Florida Law Review

Volume 71 | Issue 2 Article 1

Staking the Boundaries of Software Copyrights in the Shadow of Patents

Pamela Samuelson

STAKING THE BOUNDARIES OF SOFTWARE COPYRIGHTS IN THE SHADOW OF PATENTS

Pamela Samuelson[*]

Abstract

Ever since the venerable Supreme Court opinion in *Baker v. Selden*, courts and commentators have overwhelmingly endorsed the rule that copyright and utility patent protections for intellectual creations are mutually exclusive. That is, an intellectual creation may be eligible for copyright or utility patent protection, but not both. Original works of authorship are channeled to the copyright regime, while novel and nonobvious technologies are channeled to the patent system.

The well-established mutual exclusivity rule for copyright and utility patents was recently renounced, as applied to computer program innovations, by the Court of Appeals for the Federal Circuit (CAFC) in *Oracle America, Inc. v. Google Inc.* One of Google's defenses against Oracle's charge of copyright infringement for its reuse of parts of the Java application program interface (API) was that the API's functionality made it more properly patent, than copyright, subject matter. The CAFC rejected the exclusivity argument and endorsed overlapping copyright/patent protection for APIs.

Oracle is the latest exemplar of the vexing conceptual difficulties that computer programs have posed for both copyright and patent laws over the past fifty-some years. This Article provides an overview of the mutual exclusivity rule that grew out of *Baker*. It also explains why the CAFC's *Oracle* decision is deeply flawed, and why courts should renew their commitment to the mutual exclusivity rule in software copyright cases to ensure that, consistent with long-standing limiting principles of copyright law and Supreme Court precedents, copyright will not be construed to give patent-like protection to program functionality.

This Article also offers concrete suggestions about how courts should approach discerning the proper boundaries of copyright and patent in protecting particular aspects of software. More clarity to the software copyright caselaw can be attained if courts engage in rigorous filtration of unprotectable nonliteral elements of software. When courts interpret

* Richard M. Sherman Distinguished Professor of Law, Berkeley Law School. I thank Kathryn Hashimoto and Brookes Degen for outstanding research support and many colleagues for their comments on an earlier draft entitled "The Waxing and Waning of Copyright and Patent Protections for Software," including Clark Asay, Shyam Balganesh, Michael Barclay, Chris Buccafusco, Dan Burk, Michael Carrier, Julie Cohen, Kevin Collins, Charles Duan, Jim Gibson, John Golden, Wendy Gordon, James Grimmelmann, David Hayes, Dave Jones, Mark Lemley, Jessica Litman, Lydia Loren, Corynne McSherry, Peter Menell, Jerry Reichman, Josh Sarnoff, Jule Sigall, Chris Sprigman, Jennifer Urban, and Molly Van Houweling.

software copyright in the shadow of patents, they are less likely to exceed the boundaries articulated by *Baker* and § 102(b) of the Copyright Act of 1976.

INTRODUCTION

Ever since the venerable U.S. Supreme Court opinion in *Baker v. Selden*,[1] courts and commentators have overwhelmingly endorsed the rule that copyright and utility patent protections for intellectual creations are mutually exclusive.[2] That is, an intellectual creation may be eligible for copyright or utility patent protection, but not for both.[3] Original works of authorship have been channeled to the copyright regime, while novel and nonobvious technologies have been channeled to the patent system.[4]

The well-established mutual exclusivity rule for copyright and utility patents was recently renounced, as applied to computer program innovations, by the Court of Appeals for the Federal Circuit (CAFC) in *Oracle America, Inc. v. Google Inc*.[5] Oracle charged Google with infringing copyright by its use of 37 of the 166 Java application program interface (API) packages in its Android platform for smartphones.[6] One of Google's defenses against Oracle's charge of copyright infringement was that the functionality of the Java API made it more properly patent,

1. 101 U.S. 99 (1879).

2. For the leading cases, see *infra* Section I.B. The overwhelming majority of scholars concur that copyright and patent are and should be mutually exclusive forms of intellectual property (IP) protections. *See* Pamela Samuelson, *Strategies for Discerning the Boundaries of Copyright and Patent Protections*, 92 NOTRE DAME L. REV. 1493, 1496 n.13 (2017) (citing authorities).

3. This exclusivity traces back further to the U.S. Constitution, which authorizes Congress to enact legislation "[t]o promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their *respective* Writings and Discoveries[.]" U.S. CONST. art. I, § 8, cl. 8 (emphasis added).

4. *See* 17 U.S.C. § 102(a) (2012) (providing copyright protection for "original works of authorship"); 35 U.S.C. § 101 (2012) (providing patent protection for novel, nonobvious, and useful machines, manufactures, compositions of matter, and processes). This Article uses the word "patent" to refer to utility patents. When a context calls for reference to design patents, this Article will use that term.

5. 750 F.3d 1339, 1379–81 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

6. *Id.* at 1350–51.

rather than copyright, subject matter.[7] Google pointed to patents that Oracle and Sun had obtained on API designs in support of this view.[8] Although the trial court agreed that these patents supported Google's defense,[9] the CAFC rejected the exclusivity argument and endorsed overlapping copyright and patent protection for APIs.[10]

This Article explains why the CAFC's endorsement of such overlapping protection is deeply flawed and why courts should renew their commitment to the mutual exclusivity rule in software copyright cases to ensure that, consistent with long-standing limiting principles of copyright law and Supreme Court precedents, copyright will not be construed to give patent-like protection to program functionality. As Google has once more petitioned the Supreme Court to review this case,[11] there may be an opportunity for the Court to correct the CAFC's erroneous rulings.[12]

---

7. *See, e.g.*, Google's April 22, 2012 Copyright Liability Trial Brief at 1–3, Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. 3:10-cv-03651 WHA).

8. *See, e.g.*, Google's 4/3/12 Copyright Liability Trial Brief at 18–19, *Oracle*, 872 F. Supp. 2d 974 (No. 3:10-cv-03651 WHA). Google's main defenses were that the Java API elements it used were unprotectable systems under 17 U.S.C. § 102(b) or under the merger doctrine. *See, e.g.*, Google's 4/5/12 Copyright Liability Trial Brief at 1, 8, *Oracle*, 872 F. Supp. 2d 974 (No. 3:10-cv-03651 WHA). I discussed these defenses at length in Pamela Samuelson, *Functionality and Expression in Computer Programs: Refining the Tests for Software Copyright Infringement*, 31 BERKELEY TECH. L.J. 1215, 1237–84 (2016).

9. Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 996–98 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339, 1381 (Fed. Cir. 2014).

10. See *infra* Section III.B for an analysis of the CAFC's opinion on the copyright/patent overlap issue. After reversing the district court ruling that the parts of the Java API that Google used were unprotectable by copyright law, *Oracle*, 750 F.3d at 1354–71, the CAFC remanded the case for a retrial on Google's fair use defense, *id.* at 1377. A jury in 2016 found this defense persuasive. The CAFC, however, decided that no reasonable jury could have found fair use and ruled in favor of Oracle's appeal of the denial of its motion for a judgment notwithstanding the verdict. Oracle Am., Inc. v. Google LLC, 886 F.3d 1179, 1211 (Fed. Cir. 2018). The CAFC denied Google's petition for rehearing, and Google has petitioned the Supreme Court to review the CAFC's decisions as to both copyrightability and fair use. Petition for Writ of Certiorari, Google LLC v. Oracle Am., Inc., No. 18-956 (Jan. 24, 2019).

11. Petition for Writ of Certiorari, *supra* note 10; *see also* Alison Frankel, *Google Wants Supreme Court to Hear Oracle Copyright Case – Just Not Quite Yet*, REUTERS (Oct. 22, 2018, 3:10 PM), https://www.reuters.com/article/legal-us-otc-google/google-wants-supreme-court-to-hear-oracle-copyright-case-just-not-quite-yet-idUSKCN1MW2MM [https://perma.cc/3LZQ-HWYE].

12. The CAFC's copyrightability ruling has been subjected to considerable criticism, *see infra* notes 274–76 and accompanying text, as has its reversal of the fair use verdict. *See, e.g.*, Krista L. Cox, Oracle v. Google *Is More Evidence that the Federal Circuit Has No Business Deciding Copyright Cases*, ABOVE L. (Mar. 29, 2018, 4:02 PM), https://abovethelaw.com/2018/03/oracle-v-google-is-more-evidence-that-the-federal-circuit-has-no-business-deciding-copyright-cases/ [https://perma.cc/2WQ2-8TUJ].

Oracle is the latest exemplar of the vexing conceptual difficulties that computer programs have posed for both copyright and patent laws over the past fifty-some years. The main reason for these difficulties is that programs are virtual machines that just happen to have been constructed as texts in the context of intellectual property (IP) regimes that have historically assumed that a particular creation was either a writing or a machine, but could not be both at the same time.[13] This hybrid nature of programs has given rise to confusion and sometimes heated debate about the appropriate roles that copyright and patent law do and should play in the protection of software innovations.[14] While there is general consensus that computer program code is copyrightable but not patentable,[15] nonliteral elements of programs have seemed to some to be copyrightable and to others to be patentable.[16] Most commentators have agreed that copyright and patent are and should be mutually exclusive in the

13. *See, e.g.*, Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308, 2320–24 (1994).

14. Most of the very extensive law review literature on IP rights in computer programs from the 1980s and early 1990s focused either on copyright or patent protection and did not attempt to articulate what aspects of software should be protected by each of these laws. For a representative copyright commentary, see generally Anthony L. Clapes et al., *Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Protection for Computer Programs*, 34 UCLA L. REV. 1493 (1987); Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045 (1989); Arthur R. Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 HARV. L. REV. 977 (1993); J.H. Reichman, *Computer Programs as Applied Scientific Know-How: Implications of Copyright Protection for Commercialized University Research*, 42 VAND. L. REV. 639 (1989); Lloyd L. Weinreb, *Copyright for Functional Expression*, 111 HARV. L. REV. 1149 (1998). For a particularly useful compendium of pre-1998 software copyright caselaw, see generally DAVID L. HAYES, FENWICK & WEST LLP, A COMPREHENSIVE CURRENT ANALYSIS OF SOFTWARE "LOOK AND FEEL" PROTECTION (1998), https://www.fenwick.com/FenwickDocuments/Look_-_Feel.pdf [https://perma.cc/AF3K-KFLX]. For a representative patent commentary, see, for example, Donald S. Chisum, *The Patentability of Algorithms*, 47 U. PITT. L. REV. 959, 1019–20 (1986). Some commentators offered their perspectives on both copyright and patent protections for software. *See, e.g.*, Duncan M. Davidson, *Common Law, Uncommon Software*, 47 U. PITT. L. REV. 1037, 1054–63 (1986); Gregory J. Maier, *Software Protection—Integrating Patent, Copyright and Trade Secret Law*, 69 J. PAT. & TRADEMARK OFF. SOC'Y 151, 165 (1987); John Swinson, *Copyright or Patent or Both: An Algorithmic Approach to Computer Software Protection*, 5 HARV. J.L. & TECH. 145, 212 (1991).

15. *See, e.g.*, U.S. COPYRIGHT OFFICE & U.S. PATENT & TRADEMARK OFFICE, PATENT-COPYRIGHT LAWS OVERLAP STUDY 11 (1991) [hereinafter OVERLAP STUDY].

16. *See, e.g.*, Morton David Goldberg & John F. Burleigh, *Copyright Protection for Computer Programs: Is the Sky Falling?*, 17 AIPLA Q.J. 294, 298 (1989) (endorsing copyright protection for program structure); Steven W. Lundberg et al., Baker v. Selden*, Computer Programs, 17 U.S.C. Section 102(b) and* Whelan *Revisited*, 13 HAMLINE L. REV. 221, 250 (1990) (suggesting that program structure is patentable).

protection they provide to different aspects of programs;[17] yet, a few commentators have endorsed overlapping copyright and patent protections for specific aspects.[18]

Part I of this Article begins with a review of the mutual exclusivity rule that grew out of *Baker* and explains that the functionality and written character of programs initially caused some confusion about whether copyright or patent was an appropriate form of IP protection for programs. After Congress decided copyright was appropriate, a few courts in the mid-to-late 1980s decided that copyright protection could extend to the "structure, sequence, and organization" (SSO) of programs. These decisions were widely criticized for giving patent-like protection to programs and raising the spectre of overprotection of software by copyright that would have deleterious effects on competition and innovation in the software industry. Members of Congress were sufficiently concerned about overbroad protections that they held hearings and commissioned studies to address the perceived copyright/patent overlap problem.

---

17. *See, e.g.*, Timothy K. Armstrong, *Symbols, Systems, and Software as Intellectual Property: Time for CONTU, Part II?*, 24 MICH. TELECOMM. & TECH. L. REV. 131, 133–34 (2018); Paul Goldstein, *Infringement of Copyright in Computer Programs*, 47 U. PITT. L. REV. 1119, 1130 (1986); Dennis S. Karjala, *Distinguishing Patent and Copyright Subject Matter*, 35 CONN. L. REV. 439, 448 (2003); Dennis S. Karjala, *The Relative Roles of Patent and Copyright in the Protection of Computer Programs*, 17 J. MARSHALL J. COMP. & INFO. L. 41, 50–51 (1998) [hereinafter Karjala, *Relative Roles*]; Mark A. Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECH. L.J. 1, 25–27 (1995); David G. Luettgen, *Functional Usefulness vs. Communicative Usefulness: Thin Copyright Protection for the Nonliteral Elements of Computer Programs*, 4 TEX. INTELL. PROP. L.J. 233, 273 (1996); Steven W. Lundberg et al., *The Copyright/Patent Interface: Why the Utilitarian "Look and Feel" Is Uncopyrightable Subject Matter*, 6 COMPUTER LAW. 5, 5 (1989). The copyright/utility patent mutual exclusivity rule was questioned in the Nimmer treatise from its first edition in 1963 through the 2017 revision. *See* MELVILLE B. NIMMER, NIMMER ON COPYRIGHT § 38 (1963); 1 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 2A.07[A] (2017) [hereinafter NIMMER & NIMMER 2017]. This treatise has recently been revised and now offers a more nuanced view on the mutual exclusivity rule. *See* 1 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 2A.07[A][3] (2018) [hereinafter NIMMER & NIMMER 2018]. It acknowledges this revision was prompted in part by Samuelson, Samuelson, *supra* note 2, at 1505–12, which criticized the Nimmer treatise's copyright/patent overlap theory. *See* NIMMER & NIMMER 2018, *supra*, § 2A.07[A][4].

18. *See, e.g.*, David A. Einhorn, *Copyright and Patent Protection for Computer Software: Are They Mutually Exclusive?*, 30 IDEA 265, 266 (1990). In the mid to late 1980s, IP lawyers were split on the software IP exclusivity or overlap issue. *See, e.g.*, Pamela Samuelson, *Survey on the Patent/Copyright Interface for Computer Program*s, 17 AIPLA Q.J. 256, 260–66 (1989). In 1989, a group of IP professors was unable to reach consensus about mutual exclusivity of copyright and patent in protecting computer program innovations. *See* Donald S. Chisum et al., *LaST Frontier Conference Report on Copyright Protection of Computer Software*, 30 JURIMETRICS J. 15, 28 (1989).

Part II shows that concerns about copyright and patent overlaps subsided in the 1990s as numerous appellate court decisions recognized that the essentially utilitarian nature of programs meant that they should receive a relatively thin scope of copyright protection. These courts often acknowledged that the availability of patent protection for software innovations had some bearing on the proper scope of copyright protection for programs. Courts adapted various limiting doctrines of copyright law to prevent copyright/patent overlap in software IP cases and left it to patent law to protect programs' utilitarian elements. By interpreting the scope of copyright in the shadow of patents and filtering out functional design elements, courts in software copyright cases restored the balance that Congress intended when it excluded methods and processes from the protections afforded by copyright law.

Part III shows that the CAFC's *Oracle* decision is the sole unreversed judicial opinion to have endorsed software copyright/patent overlaps. To support its view that copyright and patent could provide overlapping protections for software APIs, the CAFC inappropriately invoked dicta from the Supreme Court's *Mazer v. Stein*[19] decision, failing to notice that the Court in *Mazer* explicitly recognized that utility patents and copyrights are "mutually exclusive."[20] Part III also urges courts to be more receptive to arguments that the existence of issued patents for software innovations is evidence that those innovations should not be protectable by copyright law, even if such patents are not conclusive evidence that the innovations are uncopyrightable.

Part IV discusses *Oracle*'s destabilization of the copyright/patent balance in software cases and offers concrete suggestions about how courts should approach discerning the proper boundaries of copyright and patent in protecting particular aspects of software. It argues that courts should be warier than they have been to date of claims of copyright in program SSO because the SSO term obscures the distinction between copyrightable expression in programs and patentable methods of operation that Congress intended to exclude from copyright's protection. More clarity in the software copyright caselaw can be attained if courts engage in rigorous filtration of unprotectable nonliteral elements of software and use computer-science-relevant terminology to describe these elements. Finally, Part V concludes.

19. 347 U.S. 201 (1954).
20. *Id.* at 215 n.33.

I. SETTING THE CONTEXT FOR UPHOLDING THE MUTUAL EXCLUSIVITY OF COPYRIGHT AND PATENT FOR COMPUTER PROGRAM INNOVATIONS

The conception of copyright and patent as mutually exclusive forms of IP rights derives from the Supreme Court's 1879 decision in *Baker v. Selden*.[21] Selden claimed copyright not only in the books he had written, but also in the bookkeeping system set forth in his books and embodied in forms he devised to instantiate the system.[22] Because the preface to Selden's book mentioned that he had sought a patent for this system, Baker argued that Selden regarded the system as patent, not copyright, subject matter.[23] The Court agreed, holding that Selden's copyright could protect his explanation of the system but did not extend to the system or the forms that instantiated that system.[24] To obtain exclusive rights in useful arts, such as a bookkeeping system, the Court said, the creator must satisfy the more rigorous qualitative standards (that is, novelty and nonobviousness, not merely originality) and examination process required by the patent system.[25] The Court sharply distinguished the writings of authors, which copyright protects, from inventions in the useful arts, which can be protected, if at all, only through the patent system (and then for a much shorter duration after a more rigorous review than copyright requires).[26]

Taylor Instrument Co. v. Fawley-Brost Co.[27] confirmed the mutual exclusivity conception of copyright and patent. In *Taylor*, the U.S. Court of Appeals for the Seventh Circuit rejected claims of copyright in temperature recording charts because although charts were statutorily eligible for copyright protection, these particular charts were essential parts of machines and the subject matter of expired utility patents.[28] The

21. 101 U.S. 99, 102 (1879).

22. *Id.* at 100. *Baker* is fundamentally a case about the distinction between the subject matters of copyright (writings of authors) and patent (inventions in the useful arts), not about the abstract idea/expression distinction. *See* Pamela Samuelson, *The Story of* Baker v. Selden*: Sharpening the Distinction Between Authorship and Invention*, *in* INTELLECTUAL PROPERTY STORIES 159, 186–87 (Jane C. Ginsburg & Rochelle Cooper Dreyfuss eds., 2006).

23. Samuelson, *supra* note 22, at 174–75.

24. *Baker*, 101 U.S. at 104–05.

25. *Id.* at 102–04.

26. Samuelson, *supra* note 2, at 1496–1503, 1512–16 (discussing constitutional, statutory, caselaw, and policy reasons why copyright and patent rights do not and should not overlap in the same intellectual creations).

27. 139 F.2d 98, 99 (7th Cir. 1943); *see also* Brown Instrument Co. v. Warner, 161 F.2d 910, 911 (D.C. Cir. 1947) (following *Taylor* in upholding the Copyright Office's refusal to register claims of copyright in temperature recording charts).

28. *Taylor*, 139 F.2d at 99–101. When patents expire, the intellectual creation is dedicated to the public domain. *See id.* at 100–01.

court conceived of copyright and patent subject matters as mutually exclusive:

> While it may be difficult to determine in which field protection must be sought, it is plain . . . that it must be in one [copyright] or the other [patent]; it cannot be found in both. In other words, there is no overlapping territory, even though the line of separation may in some instances be difficult of exact ascertainment.[29]

In keeping with *Baker*, the court regarded the utility patent regime as having supremacy over any claim of copyright in human creations that were patentable subject matters.[30] The Seventh Circuit thought it would be "intolerable" to allow Taylor to use copyright to extend its monopoly in these charts.[31] The Supreme Court's *Mazer* decision confirmed the "mutual exclusiv[ity]" of copyright and patent protections, citing favorably to *Baker*, *Taylor*, and other *Baker* progeny.[32]

Consistent with these decisions, courts should interpret the scope of copyright protection for software narrowly to avoid providing patent-like protection to functional aspects of software for which its creator did not seek a patent.[33] Copyright law should not serve as a gap-filler for functional aspects of programs for which patent protection is unavailable, either because these aspects of programs lack novelty or nonobviousness or because they are too abstract for patenting.[34] If the functional design has been patented, it should be disqualified from copyright protection.[35] Allowing otherwise would defeat an essential purpose of the patent system, which requires claimants to meet higher qualitative standards and undergo more rigorous review to acquire exclusive rights, and even then,

29. *Id.* at 99.

30. *See, e.g.*, Christopher Buccafusco & Mark A. Lemley, *Functionality Screens*, 103 VA. L. REV. 1293, 1300 (2017); Mark P. McKenna & Christopher Jon Sprigman, *What's In, What's Out: How IP's Boundary Rules Shape Innovation*, 30 HARV. J.L. & TECH. 491, 492 (2017) (discussing patent preemption of other IP rules).

31. *Taylor*, 139 F.2d at 101.

32. Mazer v. Stein, 347 U.S. 201, 215 n.33 (1954). *Mazer* and the *Oracle* decision's misinterpretation of it are discussed *infra* Section III.B.

33. *See, e.g.*, OFFICE OF TECH. ASSESSMENT, INTELLECTUAL PROPERTY RIGHTS IN AN AGE OF ELECTRONICS AND INFORMATION 83 (1986) [hereinafter 1986 OTA REPORT].

34. Since the Supreme Court's decision in *Alice Corp. v. CLS Bank Int'l*, 134 S. Ct. 2347, 2360 (2014), which struck down patent claims for a method and system for managing financial settlement risks as too abstract to be patent subject matter, it is clearer than ever that some aspects of software are too abstract to qualify for either copyright or patent protections.

35. *See infra* Part III for a discussion of this issue.

for a shorter duration.[36] Courts should not allow software developers to bypass those standards and procedures merely by asserting copyright in a software SSO innovation. It is a fundamental tenet of patent law that unpatented technological innovations are in the public domain and available for free reuses (unless they can be maintained as trade secrets).[37]

Unfortunately, applying these uncontroversial propositions in software IP cases has sometimes been difficult because the precise contours of copyrightable expression and patentable systems and processes in software are sometimes, to use *Taylor*'s words, "difficult of exact ascertainment."[38] Structural design elements of programs, such as the APIs and command structures at issue in *Oracle*, exemplify this problem. It is nevertheless important for courts to make concerted efforts to discern patent and copyright boundaries and to resist construing copyright to give patent-like protections to nonliteral elements of programs. The CAFC's broad interpretation of copyright in *Oracle*, as well as its endorsement of copyright/patent overlaps, provides fuel for undermining these fundamental norms. Articulation of these principles should inform our understanding of Congress's decision to utilize copyright as a form of protection for software and also to narrow its scope to prevent it from becoming a substitute for patenting.

A. *Overcoming Initial Conceptual Difficulties in Applying IP Rights to Software*

In the earliest days of the computer and software industries, developers created many thousands of programs without asserting either copyright or patent protections.[39] When developers wanted to claim proprietary rights in programs, they typically maintained source codes as

36. *See, e.g.*, KSR Int'l Co. v. Teleflex Inc., 550 U.S. 398, 427 (2007) ("[T]he results of ordinary innovation are not the subject of exclusive rights under the patent laws. Were it otherwise patents might stifle, rather than promote, the progress of useful arts.").

37. *See, e.g.*, Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141, 151 (1989) ("The novelty and nonobviousness requirements of patentability embody a congressional understanding, implicit in the Patent Clause itself, that free exploitation of ideas will be the rule, to which the protection of a federal patent is the exception.").

38. Taylor Instrument Co. v. Fawley-Brost Co., 139 F.2d 98, 99 (7th Cir. 1943). Discerning copyright/patent boundaries is sometimes difficult in non-software contexts as to puzzles, toys, games, and architecture. *See* Samuelson, *supra* note 2, at 1516–17; *see also* Kevin Emerson Collins, *Patent Law's Authorship Screen*, 84 U. CHI. L. REV. 1603, 1643–44 (2017) (discussing this difficulty for architectural works).

39. *See, e.g.*, MARTIN CAMPBELL-KELLY, FROM AIRLINE RESERVATIONS TO SONIC THE HEDGEHOG: A HISTORY OF THE SOFTWARE INDUSTRY 3 (2003) (discussing historical and empirical perspectives on the development of the software industry and various business models that supported it). Much software was "bundled" with hardware. *Id.* at 6. IBM's eventual "unbundling" of software and hardware under some pressure from antitrust authorities was an important development that contributed to the growth of an independent software industry. *Id.*

trade secrets and licensed the use of machine-executable codes to customers.[40] Yet, trade secrecy was, by its nature, unsuited to be the sole form of IP protection for software in the marketplace because mass-market distribution would publish the secrets.[41]

Once copyright and patent officials were confronted with claims of IP rights in programs in the mid-1960s, it became apparent that software was an awkward fit for both regimes.[42] Programs were too much of a technology to fit comfortably in the copyright regime and too much of a writing to fit comfortably in the patent system.

In 1965, the Register of Copyright decided, with some reluctance, to accept registrations for computer programs as long as their authors supplied the full texts of their source codes for deposit with the U.S. Copyright Office and complied with copyright notice requirements.[43] Those registrations were issued, however, under the Office's "rule of doubt."[44] One doubt was whether copyright protection could extend to machine-executable forms of programs, which are virtual machines and machine processes.[45] A second doubt was whether machine-executable

40. Trade secrecy and licensing remain important ways for software developers to protect their digital assets. *See, e.g.*, 1 DAVID BENDER, COMPUTER LAW: A GUIDE TO CYBERLAW & DATA PRIVACY LAW § 3.03 (2018) (including sections on trade secret and licensing agreements, among others, as types of practical and effective software protection).

41. *See, e.g.*, Videotronics, Inc. v. Bend Elecs., 564 F. Supp. 1471, 1475–78 (D. Nev. 1983) (denying trade secrecy claim for videogame program). The court noted that copyright law would have provided a more useful way to protect the plaintiff's code against unlicensed copying; trade secrecy could not, however, serve as a substitute form of IP protection after copies of the programs had circulated in the marketplace. *Id.* at 1477.

42. Justice Breyer's tenure article regarded the "case" for extending either copyright or patent protection to computer programs to be at best "uneasy" and actually quite unpersuasive. *See* Stephen Breyer, *The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs*, 84 HARV. L. REV. 281, 284 (1970). Adequate incentives for the development of computer software existed, including business models that did not depend on IP rights. *Id.* at 344–50 (describing economics of software development). See generally Pamela Samuelson, *The Uneasy Case for Software Copyrights Revisited*, 79 GEO. WASH. U. L. REV. 1746 (2011) for a contemporary assessment of Breyer's analysis.

43. COPYRIGHT OFFICE CIRCULAR 31D (Jan. 1965), *reprinted in* Duncan M. Davidson, *Protecting Computer Software: A Comprehensive Analysis*, 1983 ARIZ. ST. L.J. 611, 652 n.72 (1983).

44. Davidson, *supra* note 43, at 739. Copyright Office policy is to register a claim to copyright under the "rule of doubt" if it "has reasonable doubt as to whether the material submitted for registration constitutes copyrightable subject matter or whether the other legal and formal requirements of the statute have been met." U.S. COPYRIGHT OFFICE, COMPENDIUM OF U.S. COPYRIGHT OFFICE PRACTICES § 607 (3d ed. 2017).

45. In the first copyright case based on the copying of machine-executable code, a trial court rejected a copyright claim because the code was a "mechanical device" ineligible for copyright protection. *See* Data Cash Sys., Inc. v. JS&A Grp., Inc., 480 F. Supp. 1063, 1065–69 (N.D. Ill. 1979) (noting failure to satisfy notice of copyright requirements), *aff'd on other grounds*, 628 F.2d 1038 (7th Cir. 1980).

code could be considered a "copy" of the source code under copyright precedents.[46] Perhaps in part because of these doubts, the Office registered relatively few programs in the 1960s and 1970s.[47]

In 1967, during congressional hearings on proposed bills to overhaul U.S. copyright law,[48] one witness warned against software copyrights as likely to confer "patentlike protection under the guise of copyright."[49] Seemingly in response to those concerns, Congress added a provision to the copyright revision bill that, as enacted in the Copyright Act of 1976 (1976 Act),[50] became § 102(b): "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is . . . embodied in such work."[51] The legislative history expressly stated that it had been added to the statute to ensure that computer program copyrights would not be construed too broadly.[52]

During this same period, the U.S. Patent Office was struggling with whether to issue patents for programs and for processes embodied in programs. It perceived programs to be unpatentable printed matter (that

46. *See* White-Smith Music Publ'g Co. v. Apollo Co., 209 U.S. 1, 17–18 (1908) (stating piano rolls are not "copies" of musical compositions, but essential parts of machines that were unreadable by humans). The *Data Cash* decision analogized source code to architectural drawings and object code to buildings constructed from those drawings. *Data Cash*, 480 F. Supp. at 1068 (citing Nucor Corp. v. Tenn. Forging Steel Serv., Inc., 476 F.2d 386, 391 n.8 (8th Cir. 1973) (stating that a building is not a "copy" of drawings, but a "result" of them)). Architectural works only became copyrightable subject matter in the U.S. in 1991. Architectural Works Copyright Protection Act, Pub. L. No. 101-650, §§ 701–06, 104 Stat. 5133 (1990).

47. *See, e.g.*, NAT'L COMM'N ON NEW TECH. USES OF COPYRIGHTED WORKS, FINAL REPORT 34 (1978) [hereinafter CONTU REPORT] (reporting that IBM and Burroughs accounted for more than 80% of the 1,205 programs registered before 1977). The source code disclosure requirement may have discouraged registrations. *See, e.g.*, Davidson, *supra* note 14, at 1058. Moreover, many software developers had adopted business models not dependent on IP rights. Breyer, *supra* note 42, at 344–45.

48. *See Copyright Law Revision: Hearings on S. 597 Before the Subcomm. on Patents, Trademarks, and Copyrights of the S. Comm. on the Judiciary*, 90th Cong. 969–74 (1967), *reprinted in* 9 OMNIBUS COPYRIGHT REVISION LEGISLATIVE HISTORY 192–97 (George S. Grossman ed., 1976) [*hereinafter* 1967 Senate Hearings]; *see also* Armstrong, *supra* note 17, at 137–38 (discussing these hearings).

49. 1967 Senate Hearings, *supra* note 48, at 197 (testimony of Arthur Miller). Professor Miller also believed that Congress should not allow programmers to use copyright to protect efficient program innovations without meeting patent law's procedural or substantive standards. *Id.*

50. Pub. L. No. 94-553, 90 Stat. 2541 (codified as amended at 17 U.S.C. §§ 101–1401 (2012)).

51. 17 U.S.C. § 102(b). See generally Pamela Samuelson, *Why Copyright Law Excludes Systems and Processes from the Scope of Its Protection*, 85 TEX. L. REV. 1921 (2007), for a detailed history of this provision and the caselaw it was intended to codify. It is worth noting that processes are among the subject matters of patents. *See* 35 U.S.C. § 101 (2012).

52. H.R. REP. NO. 94-1476, at 57 (1976).

is, writings of authors) and program processes to be unpatentable mental processes.[53] The Patent Office's opposition to software patents stiffened after a presidential commission in 1966 rebuffed a proposal that program innovations be made patentable.[54] The Commission expressed concern about the Patent Office's competence to judge the patentability of software innovations because the Office lacked an appropriate database of prior art and a suitable classification system.[55] Substantial progress had been made in the programming field without patents, and besides, copyright protection was available for programs.[56]

Notwithstanding the Presidential Commission's statement, some doubts lingered about whether machine-executable forms of programs were indeed copyright-protectable, even after passage of the 1976 Act.[57] These doubts were put to rest by the National Commission on New Technological Uses of Copyrighted Works (CONTU) in the late 1970s.[58]

---

53. Any process or system for which a computer program can be written can also be instantiated in hardware. Patenting seems appropriate if one accepts, as one must, the equivalence of hardware and software and recognizes that software is a virtual machine. See generally Pamela Samuelson, *Benson Revisited: The Case Against Patent Protection for Algorithms and Other Computer Program-Related Inventions*, 39 EMORY L.J. 1025 (1990), for a more extensive treatment of the problems with software-related patents. The law review literature on software patents is vast. *See, e.g.*, James P. Chandler, *Patent Protection of Computer Programs*, 1 MINN. INTELL. PROP. REV. 33, 36–38 n.15 (2000) (reporting that over 200 articles were written about patenting of computer programs between 1970 and 1979, over 500 between 1980 and 1989, and well over 1,000 between 1990 and 2000, with citations to a substantial number of them).

54. *See* THE PRESIDENT'S COMM'N ON THE PATENT SYS., "TO PROMOTE THE PROGRESS OF . . . USEFUL ARTS" IN AN AGE OF EXPLODING TECHNOLOGY, S. DOC. NO. 91-5, at 21 (1st Sess. 1967) ("Uncertainty now exists as to whether the statute permits a valid patent to be granted on programs. Direct attempts to patent programs have been rejected on the ground of nonstatutory subject matter. Indirect attempts to obtain patents and avoid the rejection, by drafting claims as a process, or a machine or components thereof programmed in a given manner, rather than as a program itself, have confused the issue further and should not be permitted.").

55. *Id.*

56. *Id.*

57. The 1976 Act initially had a placeholder provision in 17 U.S.C. § 117 preserving the state of the law (whatever that was) as to computer software. *See* Pamela Samuelson, *CONTU Revisited*: *The Case Against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 DUKE L.J. 663, 696–98 (1984).

58. CONTU was established by Pub. L. No. 93-573, 88 Stat. 1873 (1974). *See* CONTU REPORT, *supra* note 47, app. B. It was created to address several new technology issues that had been holding up enactment of the revised copyright bill; namely, to what extent photocopying of in-copyright materials was infringement, whether inputting a copyrighted work into a computer would infringe copyright, whether databases were copyrightable, and whether computer-generated works could be copyrighted. *Id.* CONTU provided guidelines about photocopying, *id.* at 48–78, and concluded that inputting a copyrighted work into a computer could infringe copyright, *id.* at 39–40, that databases could be copyrighted if original, *id.* at 38–40, and that computer-generated works could be copyrighted if original, *id.* at 45. The copyrightability of computer programs was not in its initial charter. Weinreb, *supra* note 14, at 1165. Perhaps because

CONTU strongly endorsed copyright protection for computer programs,[59] albeit over vigorous dissents expressing concerns about copyright protecting highly functional works.[60] CONTU concluded that of the existing legal regimes, copyright was the most suitable form of IP protection.[61] It noted that Supreme Court decisions had cast doubt about the patentability of program innovations.[62] The uncertainty of patents made copyright seem like a better choice.[63] CONTU expressed confidence that courts would be able to apply traditional principles of copyright law to programs.[64] Soon after CONTU issued its report,

---

Congress had not expected CONTU to address the software issue, none of the CONTU Commissioners had expertise in computer or software technologies. *Id.* at 1166–69.

59. CONTU REPORT, *supra* note 47, at 12 (asserting that computer programs were already copyrightable under the 1976 Act). There was, however, more ambiguity on this point than CONTU acknowledged. *See* Samuelson, *supra* note 57, at 703; Armstrong, *supra* note 17, at 143–44.

60. *See* CONTU REPORT, *supra* note 47, at 27–37 (dissent of John Hersey) (arguing that programs were too functional to be copyright subject matter and failed to communicate the works' expression to humans as other copyrighted works do); *see also id.* at 37–38 (dissent of Rhoda Karpatkin).

61. *See id.* at 16–18 (comparing copyright with other existing IP regimes).

62. *Id.* at 17 (noting the Court's doubts about the patentability of program innovations). The Court unanimously ruled against the patentability of a program algorithm in *Gottschalk v. Benson*, 409 U.S. 63, 68 (1972) (holding the method for transforming binary coded decimals to pure binary form too abstract to be patentable). While CONTU was still deliberating about software protections, the Court issued another anti-software patent decision in *Parker v. Flook*, 437 U.S. 584, 594 (1978) (ruling against a claim for a software-implemented process for updating alarm limits for catalytic conversion systems). *See also* Dann v. Johnston, 425 U.S. 219, 229–30 (1976) (holding data processing claims ineligible for patenting). Not until after the 5-4 decision in *Diamond v. Diehr*, 450 U.S. 175, 191–93 (1981), which upheld a patent for a rubber-curing process that utilized a program, did patents begin to become a viable form of protection for certain aspects of software.

63. CONTU did not consider the possibility of sui generis (of its own kind) form of IP protection for software, although some had suggested this option. *See, e.g.*, World Intell. Prop. Org. [WIPO], Rep. of the Int'l Bureau on the Legal Protection of Computer Software, U.N. Doc. LPCS/1/2, at 4 (1979); Elmer Galbi, *Proposal for New Legislation to Protect Computer Programming*, 17 BULL. COPYRIGHT SOC'Y 280, 283–92 (1970); Benjamin Kaplan, *An Unhurried View of Copyright: Proposals and Prospects*, 66 COLUM. L. REV. 831, 843 (1966) ("[W]e ought to be thinking, not copyright, but patent, or perhaps a third quiddity . . . as we are told that the programs, or some of them, can be translated, so to speak, into physical parts of the computer's machinery or circuitry."). An important advantage of a copyright, rather than a sui generis, model of IP protection for software was the existence of international treaties under which software might be subsumed (as "literary works"), and hence protected internationally. Under a sui generis option, a new treaty would have had to be negotiated, which might not have been as protective of software innovations as copyright.

64. *See* CONTU REPORT, *supra* note 47, at 16. *But see infra* note 73 and accompanying text.

Congress in 1980 amended the 1976 Act to effectuate CONTU's recommendations.[65]

B. *The Road to* Whelan v. Jaslow *and Concerns About Copyright Conferring Patent-Like Protections to Software*

In the first four years after the 1980 amendments, courts rendered twenty-four software copyright decisions. Virtually all of them raised relatively straightforward issues.[66] None seemed to bear out earlier warnings about copyright giving patent-like protections to software. The infringement claims were overwhelmingly based on copying of audiovisual elements of videogames,[67] exact copying of code,[68] or both.[69]

However, in 1985, software copyright caselaw turned dicey. All five of the software copyright cases decided that year involved "nonliteral" infringement claims, that is, claims that the defendants copied structural elements of computer programs.[70] CONTU had emphasized the

65. Pub. L. No. 96-517, § 10, 94 Stat. 3015, 3028 (1980) (codified at 17 U.S.C. §§ 101, 117 (2012)) (defining "computer program" and creating a limitation enabling copying incidental to use, backup copying, necessary adaptations, and resales of one's copy). CONTU recommended that "rightful possessors" have these § 117 rights. CONTU REPORT, *supra* note 47, at 12. As enacted, only owners of copies of computer programs have these rights. 17 U.S.C. § 117. Software firms have sometimes successfully used mass-market licenses to override § 117 rights to resell software. *See, e.g.*, Vernor v. Autodesk, Inc., 621 F.3d 1102, 1115–16 (9th Cir. 2010) (enforcing a mass-market license restriction forbidding resale of software).

66. *But see* Midway Mfg. Co. v. Artic Int'l, Inc., 704 F.2d 1009, 1014 (7th Cir. 1983) (holding that the sale of chips to speed up play of another firm's videogame infringed the derivative work right); Hubco Data Prods. Corp. v. Mgmt. Assistance, Inc., 219 U.S.P.Q. (BNA) 450, 457–58 (D. Idaho 1983) (finding that altering another firm's code to enable its customers to access additional features infringed).

67. *See, e.g.*, Stern Elecs., Inc. v. Kaufman, 669 F.2d 852, 856 (2d Cir. 1982) (affirming a preliminary injunction for infringement of videogame audiovisuals); Atari, Inc. v. N. Am. Philips Consumer Elecs. Corp., 672 F.2d 607, 619–20 (7th Cir. 1982) (affirming a preliminary injunction due to the similarities in characters and total concept and feel of videogames), *superseded by statute*, FED. R. CIV. P. 52(a), *as recognized in* Scandia Down Corp. v. Euroquilt, Inc., 772 F.2d 1423, 1429 (7th Cir. 1985); Atari, Inc. v. Amusement World, Inc., 547 F. Supp. 222, 229 (D. Md. 1981) (holding that a videogame copyright was not infringed).

68. *See, e.g.*, Apple Comput., Inc. v. Formula Int'l, Inc., 725 F.2d 521, 526–27 (9th Cir. 1984) (affirming a preliminary injunction against the exact copying of Apple operating system programs), *overruled by* Flexible Lifeline Sys., Inc., v. Precision Lift, Inc. 654 F.3d 989 (9th Cir. 2011); Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240, 1254–55 (3d Cir. 1983) (reversing the denial of a preliminary injunction against the exact copying of object code); GCA Corp. v. Chance, 217 U.S.P.Q. (BNA) 718, 722–23 (N.D. Cal. 1982) (granting a preliminary injunction against the exact copying of operating system programs).

69. *See, e.g.*, Williams Elecs., Inc. v. Artic Int'l, Inc., 685 F.2d 870, 878 (3d Cir. 1982). In *Midway Mfg. Co. v. Strohon*, 564 F. Supp. 741 (N.D. Ill. 1983), the court found infringement of a copyright in the plaintiff's program code, but not in its videogame graphics. *Id.* at 754.

70. *See* Q-Co. Indus., Inc. v. Hoffman, 625 F. Supp. 608, 610 (S.D.N.Y. 1985); Williams v. Arndt, 626 F. Supp. 571, 572–73 (D. Mass. 1985); E.F. Johnson Co. v. Uniden Corp. of Am.,

importance of distinguishing copyrightable program expression and uncopyrightable program process, but had offered no guidance about how courts should go about doing so, saying only that the distinction did not "shimmer with clarity."[71]

None of these first nonliteral infringement cases made any effort to assess the applicability of the § 102(b) functionality exclusions (that is, procedure, process, system, and method of operation).[72] Courts generally sought to resolve these nonliteral software copyright infringement claims by examining them through the lens of cases involving conventional "literary works" and assuming that if the structure of novels and plays could be expressive parts of those works, then the structure of programs should also be copyright-protectable.[73]

The most troubling decision was *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*,[74] which involved competing programs to automate common functions of dental laboratories.[75] Jaslow's main

623 F. Supp. 1485, 1487 (D. Minn. 1985); SAS Inst., Inc. v. S & H Comput. Sys., Inc., 605 F. Supp. 816, 830 (M.D. Tenn. 1985); Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 609 F. Supp. 1307, 1309 (E.D. Pa. 1985), *aff'd*, 797 F.2d 1222 (3d Cir. 1986). All five involved claims of infringement for copying internal structures of software. The plaintiffs prevailed in all of these cases except *Q-Co*.

71. CONTU REPORT, *supra* note 47, at 18. Nearly forty years later, this continues to be true. *See, e.g.*, Armstrong, *supra* note 17, at 156.

72. A § 102(b) exclusion should have been dispositive in *Williams*. The copyright claim arose because Arndt developed software that implemented Williams' method for trading in commodities. The court upheld Williams' claim because the two programs contained the same set of steps (i.e., they implemented the same procedure) and produced the same results. *Williams*, 626 F. Supp. at 579. Arndt may have breached Williams' confidence and been a bit of a liar, *id.* at 573–75, 580–81, but he wrote his own program to implement the plaintiff's method. Under traditional principles of copyright law, Arndt should have prevailed in the copyright case. 1986 OTA REPORT, *supra* note 33, at 81.

73. *See, e.g.*, *SAS*, 605 F. Supp. at 829–31. The Third Circuit's decision in *Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc*., 797 F.2d 1222, 1234 (3d Cir. 1986), characterized programs as "literary works" whose structure could be as expressive as the structure of novels and plays.

74. 609 F. Supp. 1307 (E.D. Pa. 1985), *aff'd*, 797 F.2d 1222 (3d Cir. 1986).

75. *Id.* at 1309. The district court should have analyzed the § 102(b) process exclusion because similarities in the parties' programs were due to both having designed them to implement the same set of dental lab business processes. The district court characterized the copyright-protectable expression in programs as "the manner in which the program operates, controls, and regulates the computer in receiving, assembling, calculating, retaining, correlating, and producing useful information either on a screen, print-out or by audio communication." *Id.* at 1320. The Third Circuit cited approvingly to this definition of program expression. *Whelan*, 797 F.2d at 1239. Notice that if one substitutes the word "method" for "manner" in the court's statement, the sentence still makes sense. These aspects of software should have been unprotectable methods under § 102(b). The trial court thought that prospective users would perceive no differences between the Whelan and Jaslow programs. *Whelan*, 609 F. Supp. at 1322. It was as if performing the same functions in the same manner necessarily infringed. *See* 1986 OTA REPORT, *supra* note

defense was that copyright protected only program code, not program structure.[76] The U.S. Court of Appeals for the Third Circuit affirmed the lower court's finding of infringement based on similarities in the two programs' file structures, in the sequences of five subroutines, and in their "overall structure."[77] The court's decision introduced the concepts of "structure, sequence, and organization" (SSO) and the "look and feel" of programs as potentially eligible for copyright protection.[78] The court reasoned that a great deal of intellectual creativity had gone into the design of the program structures and look-and-feel and that without broad copyright protection for such program innovations there would be too little incentive to create valuable programs.[79] The court announced a test for judging nonliteral infringement in software copyright cases under which "*the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea*."[80]

Whelan precipitated numerous software copyright lawsuits involving nonliteral infringement claims. Plaintiffs in those cases relied heavily on *Whelan*'s endorsement of copyright protection for program SSO and often applied its test for infringement.[81] "Look and feel" lawsuits also proliferated, the most prominent of which were *Lotus Development Corp.*

33, at 81 (suggesting that unfair competition concerns explained *Whelan* and other nonliteral infringement cases).

76. Although the district court did not discuss Jaslow's theory of the case, it is evident in the Third Circuit's decision. *See Whelan*, 797 F.2d at 1235.

77. *Id.* at 1233–34, 1242–48. The court rejected Jaslow's argument that the file structures were like the blank forms in *Baker*. *Id.* at 1242–43. The subroutines that carried out standard business methods (for example, for close-of-day and close-of-month operations and the like), *id.* at 1245, should have been excluded from copyright under § 102(b).

78. *See id.* at 1224, 1229, 1231. The court suggested that copyright protection should be available for program "look and feel," quoting an article stating that designing look-and-feel requires considerable creativity and "often is of greater commercial value than the program code." *Id.* at 1231. "Look and feel" claims are no longer found in the software copyright caselaw.

79. *Id.* at 1237–38.

80. *Id.* at 1236.

81. *See, e.g.*, Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1176–77 (9th Cir. 1989) (affirming the grant of a preliminary injunction based on the copying of a program structure); Lotus Dev. Corp. v. Paperback Software Int'l, Inc., 740 F. Supp. 37, 52 (D. Mass. 1990) (holding that copying a spreadsheet program command structure constituted infringement, citing *Whelan*); Pearl Sys. Inc. v. Competition Elecs. Inc., 8 U.S.P.Q.2d (BNA) 1520, 1524 (S.D. Fla. 1988) (protecting SSO, following *Whelan*); Digital Commc'ns. Assocs. v. Softklone Distrib. Corp., 659 F. Supp. 449, 454–55 (N.D. Ga. 1987) (characterizing *Whelan* as "[t]he leading case addressing the extent of [copyright] protection"); Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127, 1133 (N.D. Cal. 1986) (holding that the "overall structure, sequencing, and arrangement" of a greeting card preparation program fell within the "ambit of copyright protection").

v. Paperback Software International, Inc.[82] and *Apple Computer Inc. v. Microsoft Corp.*[83]

In the mid-to-late 1980s, as the *Whelan*-inspired SSO and look-and-feel cases were being litigated,[84] the CAFC was taking an increasingly broad view of the Supreme Court's 1981 decision in *Diamond v. Diehr*[85] as having opened the door to the patenting of software innovations.[86] After *Diehr*, the Patent and Trademark Office (PTO) became more liberal in its issuance of such patents.[87] The upsurge in software patenting in the mid-to-late 1980s might have been even greater but for decisions such as *Whelan*, which "lessened the need to rely on patents as the primary legal protection for software."[88]

C. *Rising Concerns About Overbroad Copyrights and Software Patents*

By the mid-1980s, ever-growing concerns about uncertain and overbroad software copyrights and the rise of software patents among IP and computing professionals came to the attention of members of Congress. These concerns prompted a request that the Office of

82. 740 F. Supp. 37 (D. Mass. 1990).

83. 35 F.3d 1435 (9th Cir. 1994), *aff'g* 799 F. Supp. 1006 (N.D. Cal. 1992). The day after the Supreme Court denied certiorari in *Jaslow Dental Lab., Inc. v. Whelan Assocs., Inc.*, 479 U.S. 1031 (1987), Lotus filed its first "look and feel" lawsuit. Complaint, Lotus Dev. Corp. v. Paperback Software Int'l, Inc., 740 F. Supp. 37 (D. Mass. 1990) (No. 1:87CV00076). The *Apple v. Microsoft* case is discussed *infra* Section II.B.3.

84. At least one court declined to follow the *Whelan* approach to judging software copyright infringement. *See, e.g.*, Plains Cotton Coop. Ass'n v. Goodpasture Comput. Serv., Inc., 807 F.2d 1256, 1262 (5th Cir. 1987) (affirming the denial of a preliminary injunction for copyright infringement because many similarities between the parties' programs were "dictated by the externalities of the cotton market" and explaining why it declined to follow *Whelan*).

85. 450 U.S. 175 (1981).

86. The *Diehr* decision was generally viewed for some years as having opened up only a narrow window for software-related patents. *See, e.g.*, Samuelson, *supra* note 53, at 1095–96 (discussing *Diehr*). This was partly because the Supreme Court had split 5-4 on the patent claim, which involved an improved rubber curing process that used a program as one element. *Diehr*, 450 U.S. at 176; *see also* Maureen A. O'Rourke, *The Story of* Diamond v. Diehr*: Toward Patenting Software*, *in* INTELLECTUAL PROPERTY STORIES 212–13 (Jane C. Ginsburg & Rochelle Cooper Dreyfuss eds., 2006) (discussing the facts of *Diehr*).

87. *See* John T. Soma & B.F. Smith, *Software Trends: Who's Getting How Many of What? 1978 to 1987*, 71 J. PAT. & TRADEMARK OFF. SOC'Y 415, 419–20 (1989) (explaining that the average number of software patents granted between 1978 and 1981 was nine per year, which increased to thirty-four in 1984 and to sixty-six in 1987, a 470% increase over this time period); *see also* OFFICE OF TECH. ASSESSMENT, FINDING A BALANCE: COMPUTER SOFTWARE, INTELLECTUAL PROPERTY, AND THE CHALLENGE OF TECHNOLOGICAL CHANGE 55 tbl.2-1 (1992) [hereinafter 1992 OTA REPORT] (charting increases in the patenting of software inventions).

88. O'Rourke, *supra* note 86, at 214.

Technology Assessment (OTA) study software IP issues and report about these developments.[89]

In 1986, the OTA reported to Congress that "neither copyright nor patent law ha[d] successfully accommodated works of function, such as computer programs."[90] Copyright seemed to provide either "too much" or "too little" protection for program innovations.[91] If copyright protection was only available to literal code, it would be easy to rewrite the same program design in non-infringing source code, but if copyright protection "extended to the logic, design, structure, performance or even the output of the computer program," this would provide too much protection and violate the strictures of § 102(b).[92] The OTA pointed to *Whelan* and two other 1985 nonliteral infringement decisions as examples of the "too much" approach.[93] The OTA warned that "[o]verly broad copyright protection would give the owner patent-like protection over processes for a much longer duration than patent law provides," and that it would do so without satisfying the more rigorous procedural and substantive requirements of the patent system.[94]

The OTA's concerns were echoed in a large number of articles published in the late 1980s that were strongly critical of *Whelan*.[95] One article characterized the decision as "The 'First Ever' Patent Based on Copyright Relief."[96] The *Whelan* test for infringement "make[s] everything 'new' about the program susceptible to copyright protection, including any patentable ideas, without regard to the requirements of

89. *See* 1986 OTA REPORT, *supra* note 33, at ii.

90. *Id.* at 59.

91. *Id.* at 78. Breyer had also worried that copyright could provide too much or too little protection to software. Breyer, *supra* note 42, at 347–48; *see also* Samuelson et al., *supra* note 13, at 2356–61 (expressing concern about probable cycles of over- and under-protection in software copyright cases).

92. 1986 OTA REPORT, *supra* note 33, at 81.

93. *Id.*

94. *Id.* at 83.

95. *See, e.g.*, Marjorie Hope Kaufman, Pearl Systems, *Functionality, Protection for Structure, and the Boundaries of Copyright Protection: An Ad Hoc Forum*, 6 COMPUTER LAW. 1, 1–2 (1989); Menell, *supra* note 14, at 1074, 1082; Reichman, *supra* note 14, at 697, 714; D. C. Toedt, Bonito Boats *and the Primacy of the Patent System—Are There Implications for Software Functionality Copyrights?*, 6 COMPUTER LAW. 12, 12–15 (1989); Steven R. Englund, Note, *Idea, Process, or Protected Expression? Determining the Scope of Copyright Protection of the Structure of Computer Programs*, 88 MICH. L. REV. 866, 881 (1990); Peter G. Spivack, Comment, *Does Form Follow Function? The Idea/Expression Dichotomy in Copyright Protection of Software*, 35 UCLA L. REV. 723, 744 (1988).

96. John P. Sumner & Steven W. Lundberg, *Patentable Computer Program Features as Uncopyrightable Subject Matter*, 17 AIPLA Q.J. 237, 239 (1989); *see also* Michael Gemignani, *Copyright Protection: Computer-Related Dependent Works*, 15 RUTGERS COMPUTER & TECH. L.J. 383, 403–05 (1989) (criticizing *Whelan* for adopting a patent-like standard of infringement).

patentability."[97] In addition, granting protection to all program SSO would risk giving programmers patent-like protection through copyright, which could have a chilling effect on software development and ongoing innovation.[98] *Whelan* wrongly assumed that there was only one unprotectable idea per program and failed to filter, during infringement analysis, several types of unprotectable elements of programs, including efficient designs and interfaces necessary for achieving interoperability with hardware or software.[99]

Moreover, IP experts concluded that courts needed to recognize that many programming design objectives are functional, such as optimization of computing speed, efficient use of memory, input/output functioning, ease of testing, maintenance and enhancement, and the need to "achieve compatibility."[100] *Whelan* failed to recognize these design constraints and to interpret the process and method of operation exclusions required by § 102(b).[101] Software should be accorded a lesser level of copyright protection to avoid protecting its functional elements.[102] Moreover, the terms "SSO" and "look and feel" were unhelpful in distinguishing protectable and unprotectable aspects of programs.[103]

Lawyers were not the only people concerned about software IP developments. The Computer Science and Telecommunications Board of the National Academies of Sciences (NAS) hosted two forums to discuss worrisome trends in software copyright and patent law that posed risks to ongoing innovation and growth in the software industry.[104] Among other things, its report pointed out that it was not always easy to determine which law should apply to nonliteral elements of programs:

---

97. Sumner & Lundberg, *supra* note 96, at 240; *see also* Lundberg et al., *supra* note 16, at 248 ("If a statutory utility patent claim can be written on a functional, utilitarian feature specified by a computer program, the feature is by definition an idea to which copyright protection does not apply." (emphasis omitted)).

98. *See, e.g.*, David Nimmer et al., *A Structured Approach to Analyzing the Substantial Similarity of Computer Software in Copyright Infringement Cases*, 20 ARIZ. ST. L.J. 625, 630 (1988). David Nimmer is, of course, co-author (with his late father) of the Nimmer treatise. Insights of this article were carried over into the treatise. *See* 4 NIMMER & NIMMER 2018, *supra* note 17, § 13.03[F].

99. Nimmer et al., *supra* note 98, at 629–30, 639–40.

100. Chisum et al., *supra* note 18, at 19–22.

101. *Id.* at 20.

102. *Id.* at 18–19.

103. *Id.* at 20, 27.

104. *See generally* STEERING COMM. FOR INTELLECTUAL PROP. ISSUES IN SOFTWARE ET AL., INTELLECTUAL PROPERTY ISSUES IN SOFTWARE (1991) [hereinafter IP ISSUES] (discussing the changing context of the software industry and the legal issues resulting from innovation).

> [C]opyright attorneys can argue cogently that disputes over the ownership of graphical displays and the sequencing of commands—that is the look and feel of user interfaces—should be resolved in the copyright arena because the issues center on creative expression. Objecting to the subjectivity of copyright concepts, such as "look and feel" and "structure, sequence, and organization," patent attorneys argue just as persuasively that the issues can be addressed more concretely by assessing the novelty and nonobviousness of useful processes incorporated into interfaces.[105]

No consensus emerged on the appropriate roles of copyright and patent in protecting software.[106]

The swirling controversies about software IP issues induced the House Subcommittee on Courts, Intellectual Property, and the Administration of Justice to hold a set of oversight hearings about computers and intellectual property in 1989 and 1990.[107] In advance of the hearings, the OTA prepared a document about the SSO and look-and-feel lawsuits, stating, "[a]t stake in these decisions is the extent to which copyright should be interpreted to give patent-like protection, albeit for a much longer period of time and without patent's high standards for innovation and originality."[108]

At one of the hearings, two prominent software developers expressed concerns about overprotection of software by copyright law, as well as about the harm that could be done to competition and ongoing innovation in the software industry if patents came to play a more substantial role.[109] After hearing this and other testimony, Representative Robert Kastenmeier stated: "I must confess that I have difficulty comprehending the difference between where a patent might be applied for or where it is more appropriate to copyright, or maybe simultaneously apply to both, or neither."[110]

---

105. *Id.* at 62.

106. *Id.* at 62–72 (discussing different views on the patent/copyright interface, potential benefits and harms from software patents, and compatibility and standardization issues).

107. *See generally Computers and Intellectual Property: Hearings Before the Subcomm. on Courts, Intellectual Prop. & the Admin. of Justice of the H. Comm. on the Judiciary*, 102d Cong. (1989 & 1990) [hereinafter House Hearings] (conducting an oversight hearing to address challenges in computers and intellectual property).

108. *Id.* at 23. OTA posed questions that the oversight hearings should address. *Id.* at 24–25. I testified at that hearing, raising concerns from IP lawyers about overlapping copyright and patent protections for software. *See id.* at 92–95 (testimony of Pamela Samuelson).

109. *See id.* at 223–36 (testimony of Daniel S. Bricklin); *id.* at 238–43 (testimony of Mitchell D. Kapor).

110. *Id.* at 247.

The hearings raised sufficient alarm about overlapping copyright and patent protections for software to convince the congressional committees tasked with overseeing IP legislation to ask the PTO and the Copyright Office to prepare a joint study about whether patent and copyright provided overlapping protections for software.[111] In addition, the OTA prepared a follow-up report that expressed continuing concern about the software copyright caselaw, especially the controversies about copyright claims for SSO and interfaces.[112] The report noted that no consensus about how to distinguish copyrightable expression and uncopyrightable processes had yet emerged, and that some copyright cases seemed to extend protection to program functionality, contrary to the strictures of § 102(b).[113] Because courts were stretching copyright to protect nonliteral aspects of programs, the OTA identified options that Congress should consider, including further amendments to the 1976 Act and creation of a sui generis regime for software to displace or complement copyright.[114]

Interest in creating a sui generis regime for software diminished in the 1990s as courts came to conceive of programs as highly utilitarian, and of their utilitarian elements as more suitable for patent than copyright protections. To avoid *Whelan*-like overprotection, courts developed a set of doctrinal strategies to ensure that copyright law would not confer patent-like protections on program innovations, which warded off the overprotection problem—at least until the CAFC's 2014 *Oracle* decision.

II. TAKING THE UTILITARIAN NATURE OF PROGRAMS AND AVAILABILITY OF PATENTS INTO ACCOUNT IN SOFTWARE COPYRIGHT CASES

The tide turned against overbroad copyright decisions with the Second Circuit's landmark decision in *Computer Associates International, Inc. v. Altai, Inc*.[115] *Altai* recognized that the utilitarian nature of computer programs gave rise to a need to filter out many types of unprotectable elements before proceeding to judge copyright infringement in software cases.[116] After *Altai*, appellate courts in many circuits began to take into

111. *See* OVERLAP STUDY, *supra* note 15, at 1–2. This resulted in a 1991 report saying that there was no overlapping terrain in the legal protections that copyright and patent provided to software. It concluded that copyright protected program expression, and patent law protected program functions. *See id.* at 90. For more discussion of this study, see *infra* notes 148–50, 175–77 and accompanying text.

112. *See* 1992 OTA REPORT, *supra* note 87, at 16–23, 69–73.

113. *Id.* at 9–10, 22.

114. *Id.* at 29–31. There was robust interest among scholars in the sui generis option in the late 1980s and early to mid-1990s. *See* Samuelson et al., *supra* note 13, at 2312 n.6 (citing to the ample scholarly literature about sui generis software IP).

115. 982 F.2d 693 (2d Cir. 1992); *see* Samuelson et al., *supra* note 13, at 2359.

116. Samuelson et al., *supra* note 13, at 2359–61.

account that some nonliteral elements of programs might be patentable as a reason for copyright law to abjure protection of them.[117] Although the doctrinal bases on which these decisions rested varied somewhat, the conclusions they reached were consistent: Copyright law should be construed narrowly to avoid conferring patent-like protections on functional design elements of programs.

A. Altai *Narrowed Copyright Scope to Avoid Protecting Utilitarian Elements of Programs*

Shortly after the OTA published its second report on software IP issues, the Second Circuit rendered its decision in *Altai*.[118] Computer Associates claimed that Altai infringed by copying SSO designed to allow scheduling software to be compatible with IBM operating systems.[119] The Second Circuit heavily criticized *Whelan* and characterized its test for software copyright infringement as inconsistent with traditional principles of copyright law, as resting on a flawed understanding of computer science, and as relying too heavily on "metaphysical distinctions," failing to "place enough emphasis on practical considerations."[120]

Altai invoked *Baker* as the "doctrinal starting point" of analysis in software and other utilitarian work cases.[121] As "compared to aesthetic works, computer programs hover even more closely to the elusive boundary line described in § 102(b)."[122] While broad copyright protection for software might seem "attractive from a pure policy perspective," adopting it would have "a corrosive effect on certain fundamental tenets of copyright doctrine."[123]

---

117. *See, e.g.*, Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CALIF. L. REV. 1, 27 (2001) ("As patent and copyright law overlap more and more, it becomes critical that they take account of each other.").

118. *Altai*, 982 F.2d 693. One of my briefs played a small role in this case. *See* Pamela Samuelson et al., Brief Amicus Curiae of Copyright Law Professors, Lotus Dev. Corp. v. Borland Int'l, Inc., 799 F. Supp. 203 (D. Mass. 1992) (No. 1:90-CV-11662). It was appended to Altai's brief to the Second Circuit. *See* Brief of Defendant-Appellee at 13 n.3, *Altai*, 982 F.2d 693 (No. 91-7893).

119. *Altai*, 982 F.2d at 698, 701.

120. *Id.* at 705–06. The Second Circuit noted that *Whelan* had met with a "mixed reception" in the caselaw and "ha[d] fared even more poorly in the academic community," citing to several articles critical of *Whelan* for interpreting copyright too broadly. *Id.* at 705.

121. *Id.* at 704.

122. *Id.*

123. *Id.* at 712. The court noted that the Supreme Court's decision in *Feist Publ'ns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340 (1991), had undercut *Whelan*'s incentive-based argument for broad software copyright protection. *Altai*, 982 F.2d at 711–12.

To be true to these tenets in software copyright cases, *Altai* endorsed a three-step "abstraction, filtration, and comparison" (AFC) test for judging nonliteral infringement of software copyrights.[124] The first step involves creation of a hierarchy of abstractions of the plaintiff's program; the second step filters out unprotectable elements; the third step compares the remaining expressive elements of the plaintiff's program with the defendant's program to determine if the defendant's program is substantially similar to expressive elements copied from the plaintiff's program.[125] Among the unprotectables are those elements dictated by efficiency, those constrained by external factors, such as the need to be compatible with hardware or software, and those in the public domain.[126] The court agreed with the district court that the similarities between Altai's and Computer Associates' programs were constrained by external factors, namely, the need to be compatible with IBM programs.[127]

By 1994, the *Altai* approach to analyzing the scope of software copyrights had been endorsed by the U.S. Courts of Appeals for the Fifth, Ninth and Tenth Circuits,[128] and was well on its way to becoming the preeminent software copyright decision,[129] which it remains to this day.[130] Although the Second Circuit gave little attention to the role that patent law did or should play in protecting software innovations, it recognized that patents might protect some software innovations.[131] *Altai*'s emphasis on the highly utilitarian nature of programs at least implicitly suggests that patents might be more suitable than copyright to protect certain nonliteral elements of programs.[132]

---

124. *Altai*, 982 F.2d at 706–11.

125. *Id.*

126. *Id.* at 707–10.

127. *Id.* at 714–15.

128. *See, e.g.*, Apple Comput., Inc. v. Microsoft Corp., 35 F.3d 1435, 1442, 1445 (9th Cir. 1994) (endorsing "thin" protection for software and filtration of many types of unprotectable elements, citing to *Altai* as "relying in part on our own approach"); Eng'g Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1342 (5th Cir. 1994) (endorsing and applying the AFC test); Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 841–42 (10th Cir. 1993) (citing approvingly to *Altai* and applying the AFC test); Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524 (9th Cir. 1992) (citing approvingly to *Altai* and its recognition that efficiency, external factors, and compatibility considerations constrained program design).

129. *See, e.g.*, Lemley, *supra* note 17, at 14–15 (characterizing *Altai* as the leading case on software copyright infringement).

130. *See, e.g.*, Samuelson, *supra* note 8, at 1296. *Altai* is the most cited of the software copyright cases.

131. *Altai*, 982 F.2d at 712 (citing Randell M. Whitmeyer, Comment, *A Plea for Due Processes: Defining the Proper Scope of Patent Protection for Computer Software*, 85 NW. U. L. REV. 1103, 1123–25 (1991)).

132. The Second Circuit in *Altai* wisely focused on applying traditional limiting principles of copyright law to software rather than trying to define the boundaries of copyright protection

By construing the scope of copyright narrowly and insisting on filtration of efficient and other functional elements, *Altai* may have inadvertently contributed to a substantial increase in patenting of software-related inventions.[133] As courts across the country adopted the AFC test, it became clear that copyright was going to provide a much "thinner" scope of protection to computer programs than during the *Whelan* heyday.[134] This created incentives for software developers to seek and obtain software-related patents. Under the CAFC's 1990s decisions in *In re Alappat*[135] and *State Street Bank & Trust Co. v. Signature Financial Group, Inc.*,[136] it became easier than ever to patent software innovations as long as the claimed inventions produced a "useful, concrete, and tangible result."[137] Software patents thus became a much more salient factor in the software IP landscape during the 1990s and 2000s.[138]

B. *Doctrinal Strategies for Averting Overlapping Copyright and Patent Protections for Software*

After *Altai*, appellate courts became more attentive to differentiating the roles of copyright and patent in protecting software innovation. These cases explicitly recognized the need to narrowly construe the scope of copyright in programs to avoid providing patent-like protection to

---

for software in relation to the boundaries of patents. Courts should recognize that both laws may have a role to play in protecting programs, but copyright and patent are not co-dependent such that the boundaries of one shape the boundaries of the other.

133. *See, e.g.*, Lemley, *supra* note 17, at 26 ("As software patents gain[ed] increasingly broad protection, whatever reasons there once were for broad copyright protection of computer programs [had] disappear[ed]." (footnote omitted)).

134. *See, e.g.*, Josh Lerner & Feng Zhu, *What Is the Impact of Software Patent Shifts?: Evidence from* Lotus v. Borland 26 (Nat'l Bureau of Econ. Research, Working Paper No. 11168, 2005) (presenting evidence of a surge in patenting of software innovations in the mid-1990s). Although Lerner & Zhu attribute thin copyright protection to *Borland*, *Altai* was the more significant precedent.

135. 33 F.3d 1526 (Fed. Cir. 1994), *abrogated by In re* Bilski, 545 F.3d 943.

136. 149 F.3d 1368 (Fed. Cir. 1998), *abrogated by In re* Bilski, 545 F.3d 943.

137. *State St.*, 149 F.3d at 1370 (upholding patent claims for a software-implemented financial services invention); *In re Alappat*, 33 F.3d at 1544–45 (upholding a patent claim for a software machine that smoothed wave forms for oscilloscopes).

138. *See, e.g.*, O'Rourke, *supra* note 86, at 216–18 (discussing the importance of *Alappat* and *State Street* for software-related patents). Not until 2010 did patentable subject matter become a renewed basis on which to strike down software-related patents. Bilski v. Kappos, 561 U.S. 593 (2010). In *Bilski*, the Supreme Court ruled that a method of hedging the risk of price fluctuations of commodities was too abstract to be patentable. *Id.* at 611–12. *Bilski* presaged the eventual narrowing of patent subject matter for software-related innovations, as happened in the Court's 2014 *Alice* decision. *See generally, e.g.*, Jasper L. Tran, *Software Patents: A One-Year Review of* Alice v. CLS Bank, 97 J. PAT. & TRADEMARK OFF. SOC'Y 532 (2015) (discussing the impact of *Alice* on the validity of many software-related patents).

program innovations through copyright.[139] This Section identifies five doctrinal strategies that courts have adopted to avoid software copyright/patent overlaps. Each strategy is a variation on the general theme of distinguishing copyright and patent subject matters that can be traced back to *Baker*. While it is useful to identify each of these doctrinal strands, there are fewer substantive differences than this Article's categorization might suggest. In numerous cases, the courts invoked more than one of these strategies.[140]

1. The Layering Approach

Baker was the first decision to conceptualize copyright and patent as having distinct roles in protecting different intellectual creations.[141] Copyright could provide one layer of protection for some aspects of original works of authorship (for example, Selden's explanation of his bookkeeping system), but to get exclusive rights for a different layer (for example, the bookkeeping system embodied in Selden's book) one must comply with rules of the patent system which regulates functional creativity in a different way than copyright regulates expressive or aesthetic creativity.[142] This layering approach is evident in the software caselaw.

The CAFC was an early adopter of the concept of copyright and patent as playing different roles in the protection of software innovations in *Atari Games Corp. v. Nintendo of America, Inc.*[143] In that decision, which upheld a preliminary injunction forbidding Atari from copying more of Nintendo programs than was necessary to achieve compatibility with the

139. Most strategies discussed herein have counterparts to those used to discern copyright/patent boundaries in non-software cases. *See* Samuelson, *supra* note 2, at 1494. The only strategy from the non-software cases lacking a counterpart in the software context is election of protection. *Id.* at 1508. The legal and policy arguments against copyright/patent overlaps developed in that article, *id.* at 1494–99, 1512–16, apply to software as well, although the boundaries of copyright and patent to software innovations are sometimes more difficult to discern than with non-software creations. *See, e.g.*, 1986 OTA REPORT, *supra* note 33, at 59, 78–83.

140. *See, e.g.*, Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 836–38, 849 (10th Cir. 1993) (remanding a software copyright case so the trial court could filter out unprotectable processes, merged elements, and *scènes à faire* elements).

141. Baker v. Selden, 101 U.S. 99, 102 (1879); *supra* notes 21–26 and accompanying text.

142. *See* Samuelson, *supra* note 2, at 1517–21 (illustrating this strategy in non-software cases).

143. 975 F.2d 832, 835 (Fed. Cir. 1992). Nintendo sued Atari for copying a data stream necessary to enable videogames to run on its platform. *Id.* at 836–37. Atari argued that this copying was justified because the digital stream was necessary for its videogames to be compatible with the Nintendo system. *Id.* at 845. The Atari program was written in a different programming language and ran on a different processor. *Id.* at 836.

latter's platform,[144] the CAFC characterized copyright and patent laws as protecting "distinct aspects" of software.[145] The role of copyright, it said, was to protect program expression, not methods or processes embedded in software, which might well be patentable.[146] Moreover, programmers could not get patent-like protection for program methods and processes simply by embodying them in unintelligible forms (that is, object code) and then claiming copyright infringement when someone tried to understand the code (that is, by reverse engineering it to produce an approximation of the source code).[147]

The CAFC's conclusion that copyright and patent protect distinct aspects of software was consistent with the PTO-Copyright Office Overlap Study's main conclusion that there was "no overlap in subject matter: copyright protects the authorship in a set of statements that bring about a certain result in the operation of a computer, and patents cover

144. *Id.* at 845. For a critical analysis of *Atari*, see, for example, Julie E. Cohen, *Reverse Engineering and the Rise of Electronic Vigilantism: Intellectual Property Implications of "Lock-Out" Programs*, 68 S. CAL. L. REV. 1091, 1135, 1146 (1995). Like Professor Cohen, I have been more sympathetic to Atari's defense than was the CAFC. Consider its statement that "Nintendo chose *arbitrary programming instructions* and arranged them in a unique sequence to create a *purely arbitrary* data stream" that unlocked the Nintendo console. *Atari*, 975 F.2d at 840 (emphasis added). The CAFC asserted that "Nintendo may protect this creative element of the 10NES under copyright." *Id.* Yet courts have denied copyright claims because arbitrary choices do not satisfy copyright's originality requirement. *See, e.g.*, Toro Co. v. R & R Prods. Co., 787 F.2d 1208, 1213 (8th Cir. 1986) (holding that numbers assigned to lawnmower parts were not copyrightable because they were arbitrary).

Atari definitely hurt its case by lying to the Copyright Office when its lawyer asked for a copy of the registered Nintendo source code, saying it was needed to defend against an infringement claim (There was, at the time, no pending lawsuit). *Atari*, 975 F.2d at 836. Atari used the source code to discern how to make its programs run on the Nintendo machines. *Id.* The CAFC ruled that Atari's reproduction of the unauthorized copy obtained from the Copyright Office constituted infringement. *Id.* at 841–42. The lie also undermined Atari's fair use and misuse defenses. *Id.* at 843–44, 847. Eventually the litigants settled this dispute. Cohen, 68 S. CAL. L. REV. at 1104. *Atari* is yet another example of a software copyright case decided more on unfair competition grounds than on traditional scope of copyright protection grounds.

145. *Atari*, 975 F.2d at 839; *see also* Incredible Techs., Inc. v. Virtual Techs., Inc., 400 F.3d 1007, 1012 (7th Cir. 2005) (asserting that while the functional features of a game console were uncopyrightable, they might be patentable); MiTek Holdings, Inc. v. Arce Eng'g Co., 89 F.3d 1548, 1556–57 n.19 (11th Cir. 1996) (citing approvingly to *Atari* regarding copyright and patent protection of "distinct aspects" of software); Comput. Assoc. Int'l Inc. v Altai, Inc., 775 F. Supp. 544, 558, 560 (E.D.N.Y. 1991) (finding that copyright protects program expression; methods and processes are better protected by patent than copyright), *aff'd*, 982 F.2d 693 (2d Cir. 1992).

146. *Atari*, 975 F.2d at 838–39. The CAFC invoked § 102(b) in support of this proposition. *Id.*

147. *Id.* at 842; *see also* Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1525–26 (9th Cir. 1992) (finding that the use of copyrighted computer work to gain an understanding of unprotected functional elements was fair use).

novel and nonobvious computer processes."[148] Copyright and patent were available to protect "totally different aspects" of program innovations.[149] The Study cited the Supreme Court's *Baker* decision as the "bedrock opinion for the view that patent and copyright are mutually exclusive."[150] The Study thus avoided the overlap problem by endorsing a layering approach to keep the boundaries of each law separate in protecting software innovations.

2. The § 102(b) Exclusion Approach

Consistent with the layering approach, and indeed often overlapping with it, are cases in which courts have explicitly invoked § 102(b)'s exclusion of methods and processes as a way to avoid copyright/patent overlaps in software cases.[151] *Lotus Development Corp. v. Borland International, Inc.*[152] is the best known of the software copyright cases in which the § 102(b) method exclusion was outcome-determinative.[153] Under this approach, Borland was free to reuse the command hierarchy of Lotus 1-2-3 in its spreadsheet program because that hierarchy was a constituent part of an unprotectable method of operating a spreadsheet program.[154] Borland's briefs had highlighted § 102(b) as a mechanism for policing the boundaries of copyright and patent in the protection of software innovations.[155] However, the U.S. Court of Appeals for the First Circuit did not tie the rationale for the § 102(b) method exclusion to

148. Letter from Ralph Oman, Assoc. Librarian & Register of Copyrights, & Harry F. Manbeck, Assistant Sec'y & Comm'r of Patents & Trademarks, to the Hon. William J. Hughes, Chairmain, Subcomm. on Intellectual Prop. & Judicial Admin. of the Comm. on the Judiciary (July 17, 1991), *in* OVERLAP STUDY, *supra* note 15 (transmitting the Overlap Study to the Chair of the House Subcommittee). The Study reported that "[p]atent protection is not available for computer programs *per se*" because they "consist of mental steps or printed matter." OVERLAP STUDY, *supra* note 15, at iii, vii. It questioned the patentability of a "mere display on a screen of commands, menus, questions and answers, forms, or icons" because these user interface elements are "generally considered to be merely printed matter." *Id.* at 45–46. Yet, processes to generate user interfaces might be eligible for patenting. *Id.* at 47. The Study discussed the possibility of design patent protection for icons. *Id.* at 46–47.

149. OVERLAP STUDY, *supra* note 15, at 2.

150. *Id.* at 19 (citing Baker v. Selden, 101 U.S. 99, 107 (1879)).

151. The § 102(b) exclusion approach differs somewhat from the layering approach insofar as courts tie its process and method exclusions directly to patentability considerations. 17 U.S.C. § 120(b) (2012). *Lotus Dev. Corp. v. Borland Int'l, Inc.*, can, for instance, be conceptualized as a layering decision, but it did not tie the exclusion to patentable subject matter. 49 F.3d 807, 818 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

152. 49 F.3d 807 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

153. *Id.* at 818.

154. *Id.* at 816–17.

155. *See* Reply Brief of Defendant/Appellant Borland Int'l, Inc. at 42–50, *Borland*, 49 F.3d 807 (No. 93-2214); *see also* Brief for Respondent at 22–46, *Borland*, 516 U.S. 233 (No. 94-2003) [hereinafter Borland Brief] (making a similar argument in its Supreme Court brief).

concerns about copyright being construed to give patent-like protections to functional aspects of software.[156] For this reason, *Borland* is not the best illustration of the § 102(b) exclusion approach to discerning the copyright/patent boundaries.

A better example is *MiTek Holdings, Inc. v. Arce Engineering Co.*,[157] in which the Eleventh Circuit Court of Appeals applied a § 102(b) process exclusion, emphasizing the connection between the § 102(b) process exclusion and the role of patents.[158] MiTek and Arce were competitors in the market for wood truss drafting software.[159] Due to numerous similarities between the two programs, especially their command structures, MiTek sued Arce for nonliteral copyright infringement.[160]

The district court ruled that the MiTek program's menu command tree was an unprotectable process under § 102(b) because it mimicked the steps that a draftsman would typically follow in the process of designing trusses to support roofs for specific buildings.[161] The Eleventh Circuit affirmed, in part relying on § 102(b):[162]

> [T]he idea of closely correlating the ACES program to the longhand steps taken by a draftsman was the constraining force in the design of the menu and submenu command tree structure. The logic inherent in this step-by-step process renders the resulting program unoriginal in that such logic may be expressed in only a limited number of ways. More than a minor departure from the logical sequence renders the result unusable.[163]

In a long footnote, the Eleventh Circuit opined that "[w]ere we to grant copyright protection to MiTek's user interface, which is nothing more

156. Judge Boudin recognized that enforcing Lotus' copyright against Borland might give Lotus patent-like protection. *Borland*, 49 F.3d at 819 (Boudin, J., concurring). However, he did not tie this concern to § 102(b).

157. 89 F.3d 1548 (11th Cir. 1996).

158. *Id.* at 1556.

159. Wood trusses are internal supports for the roofs of buildings. They are often designed off-site by engineers who work with site-specifications. The trusses are then constructed off-site and later brought to the building site for installation. *Id.* at 1551.

160. *Id.* at 1550–51.

161. *Id.* at 1556.

162. *Id.* at 1557.

163. *Id.* at 1558. The court relied on unoriginality and merger as well as § 102(b). *Id.* at 1557 n.20. The court also rejected MiTek's theory that the command structure was a protectable compilation. *Id.* The virtual identicality standard for copyright infringement of software user interfaces was not met in *MiTek*. *Id.* at 1558–59.

than a process, we would be affording copyright protection to a process that is the province of patent law."[164]

The functionality exclusions of § 102(b) and concerns about copyright not providing patent-like protections were also significant factors in the Ninth Circuit's decision in *Sega Enterprises Ltd. v. Accolade, Inc*.[165] There, Sega sued Accolade for copyright infringement because Accolade made copies of Sega programs in reverse-engineering them to discern information about how to make its games successfully interoperate with the then-popular Genesis platform.[166] Accolade then rewrote its videogames so that they too would run on that platform.[167]

Sega did not claim infringement based on Accolade's reimplementation of the Sega interface in independently written software.[168] The Ninth Circuit's *Accolade* decision nonetheless ruled that the Sega interface was an unprotectable procedure under § 102(b).[169] The § 102(b) exclusion was relevant to Accolade's fair use defense because it reinforced Accolade's argument that its purpose in making the reverse-engineering copies was "to study the functional requirements for Genesis compatibility so that it could modify existing games and make them

164. *Id.* at 1556 n.19. Under the Supreme Court's *Alice* decision, discussed *infra* notes 309–12 and accompanying text, computerizing a wood truss drafting process would probably be ineligible for patenting. *See* Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1541 n.21 (11th Cir. 1996) ("It is particularly important to exclude methods of operation and processes from the scope of copyright in computer programs because much of the contents of computer programs is patentable. Were we to permit an author to claim copyright protection for those elements of the work that should be the province of patent law, we would be undermining the competitive principles that are fundamental to the patent system.").

165. 977 F.2d 1510, 1518, 1530 (9th Cir. 1992).

166. *Id.* at 1514–16.

167. *Id.* at 1515.

168. *See, e.g.*, Secure Servs. Tech., Inc. v. Time & Space Processing, Inc., 722 F. Supp. 1354, 1361 (E.D. Va. 1989) (suggesting that compatibility defenses might succeed as long as the second-comer had written its own implementation of a program interface); NEC Corp. v. Intel Corp., No. C-84-20799-WPG, 1989 WL 67434, at *2 (N.D. Cal. Feb. 6, 1989). The goal of Sega's lawsuit was to stop Accolade (and others) from reverse-engineering its programs. *See, e.g.*, IP ISSUES, *supra* note 104, at 77–78 (objecting to software reverse-engineering because it could be used to disguise infringement); Davidson, *supra* note 14, at 1093–99 (arguing that reverse-engineer copying was copyright infringement).

169. *Accolade*, 977 F.2d at 1522–23. The Ninth Circuit noted that "the functional requirements for compatibility with the Genesis console" were unprotectable by copyright, citing § 102(b). *Id.* at 1522. These were the "interface procedures" for the Sega console that Accolade had used when it "wrote its own procedures based on what it had learned through disassembly." *Id.* The court thus conceived of the Sega interface as an unprotectable procedure under § 102(b). *Id.* at 1522–23.

usable with the Genesis console."[170] The court perceived there to be no other way for Accolade to get this information.[171]

The Ninth Circuit recognized that ruling in favor of Sega's claim that disassembly of its programs constituted infringement would, in effect, grant Sega a "de facto monopoly over the functional aspects of [its] work—aspects that were expressly denied copyright protection by Congress," citing to § 102(b).[172] The only way that Sega could obtain exclusive rights over an interface necessary for interoperability would be by "satisfy[ing] the more stringent standards imposed by the patent laws."[173] The Ninth Circuit considered Accolade's reverse-engineering to have resulted in a public benefit because there were now more independently written programs available to owners of Genesis platforms.[174]

The *MiTek* and *Accolade* (Sega) holdings were consistent with the Overlap Study, which observed that underlying the debate over the proper scope of copyright in software was "the question of protection of functionality."[175] The Study opined that it would be contrary to the statutory exclusions set forth in 17 U.S.C. § 102(b) for copyright to protect program functionality.[176] Protection of functionality "is assigned to patents where a much more rigorous test must be undergone and the barriers to entry, in terms of time, cost, and complexity, are higher."[177] The Study also adhered to the Supreme Court's pronouncements in *Baker* and *Mazer* that copyright and patent were mutually exclusive.[178]

---

170. *Id.* "Research" is one of the statutorily favored fair use purposes. 17 U.S.C. § 107 (2012). Accolade's research purpose supported its fair use defense. *Accolade*, 977 F.2d at 1521–22.

171. *Accolade*, 977 F.2d at 1522–23. Accolade could have licensed the interface information from Sega, but decided against doing so because taking the license would have foreclosed Accolade's development of videogames for other platforms. *Id.* at 1514.

172. *Id.* at 1526 (citing 17 U.S.C. § 102(b)).

173. *Id.* The court cited to the Supreme Court's decision in *Bonito Boats, Inc. v. Thunder Craft Boats, Inc.*, 489 U.S. 141, 159–64 (1989), in support of this proposition and noted that Sega did not have a patent on the Genesis console. *Accolade*, 977 F.2d at 1526.

174. *Id.* at 1523. Sega was a large Japan-based information technology (IT) company, and Accolade was a U.S.-based startup. *See, e.g.*, Andrew Pollack, *"Fifth Generation" Became Japan's Lost Generation*, N.Y. TIMES (June 5, 1992), http://www.nytimes.com/1992/06/05/business/fifth-generation-became-japan-s-lost-generation.html [https://perma.cc/RDY7-A2HF] (describing the U.S.'s considerable anxiety in the 1980s and early 1990s that Japanese companies would dominate the world IT industry). Accolade's victory enabled American startups to compete in the videogame market.

175. OVERLAP STUDY, *supra* note 15, at 87.

176. *Id.*

177. *Id.* at 88.

178. *See id.* at iii.

3. The Thin Protection Approach

Consistent with the layering and § 102(b) strategies, but different in emphasis, have been those decisions that adopted a "thin" copyright approach to judging infringement in software copyright cases as a way to ensure that copyright would not protect functional aspects of software. The first appellate court decision to adopt this approach was the Ninth Circuit in *Apple Computer, Inc. v. Microsoft Corp.*[179] Apple claimed that Microsoft infringed Apple's copyright in its graphical user interface (GUI) because the Windows GUI was substantially similar to the creative "look and feel" of the Apple GUI.[180] The Ninth Circuit agreed with the district court that the Apple GUI was entitled to only "thin" protection.[181] Under the thin protection doctrine, infringement could be found only if the Microsoft GUI was virtually identical to Apple's.[182]

The district court explained that a "thin" protection approach was appropriate because many aspects of the Apple GUI were functional:

> Purely functional items or an arrangement of them for functional purposes are wholly beyond the realm of copyright as are other common examples of user interfaces or arrangements of their individual elements—the dials, knobs, and remote control devices of a television or VCR, or the buttons and clocks of an oven or stove.[183]

---

179. Apple Comput., Inc. v. Microsoft Corp., 35 F.3d 1435, 1446–47 (9th Cir. 1994), *aff'g* 799 F. Supp. 1006 (N.D. Cal. 1992). The Ninth Circuit drew upon earlier caselaw holding that certain audiovisual elements of videogames should be subject to "thin" protection and a test of "virtually identical" copying. *See, e.g.*, Frybarger v. Int'l Bus. Mach. Corp., 812 F.2d 525, 530 (9th Cir. 1987) ("[I]ndispensable expression . . . based on the technical requirements of the videogame medium, may be protected only against virtually identical copying." (emphasis omitted)); Atari, Inc. v. N. Am. Philips Consumer Elecs. Corp., 672 F.2d 607, 617 (7th Cir. 1982), *superseded by statute*, FED. R. CIV. P. 52(a), *as recognized in* Scandia Down Corp. v. Euroquilt, Inc., 772 F.2d 1423, 1429 (7th Cir. 1985) (finding that some expressive elements of the PAC-MAN videogame were *scènes à faire* and only infringed by virtually identical copying); *see also* Telemktg. Res. v. Symantec Corp., 12 U.S.P.Q.2d 1991, 1994–96 (N.D. Cal. 1989), *aff'd in part sub nom.* Brown Bag Software v. Symantec Corp., 960 F.2d 1465 (9th Cir. 1992) (applying analytic dissection and virtually identical standard to find defendant's program not similar, citing *Frybarger*). The thin protection approach to managing copyright/patent boundaries has also been used in non-software contexts. Samuelson, *supra* note 2, at 1533–35.

180. *Microsoft*, 35 F.3d at 1442. Apple argued that Microsoft's GUI "virtually mimicked the composition, organization, arrangement and dynamics of the Macintosh interface, as shown by striking similarities in the animation of overlapping windows and the design, layout and animation of icons." *Id.*

181. *Id.* at 1439.

182. *Id.* at 1439–40. The Ninth Circuit affirmed the trial court's grant of summary judgment to Microsoft because Apple was unwilling to go to trial under such a test. *Id.*

183. Apple Comput., Inc. v. Microsoft Corp., 799 F. Supp. 1006, 1023 (N.D. Cal. 1992), *aff'd*, 35 F.3d 1435; *see also* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 817 (1st Cir.

The user interface of a computer program was no less functional than those user interfaces.[184] Similarities in functional elements of user interfaces or arrangements of them, said the district court, "does not suggest unlawful copying, but standardization across competing products for functional considerations."[185]

The Ninth Circuit concurred in this "thin" protection approach and held that the district court had rightly applied "limiting doctrines of originality, functionality, standardization, *scenes a faire*, and merger" to each allegedly infringing element of the Apple GUI.[186] Apple had, for example, designed its GUI so a cursor could be used to move a document icon to a trash can icon instead of using the delete key to dispose of unwanted items.[187] The Ninth Circuit thought that this "exemplifie[d] an essentially functional process, indispensable to the idea of manipulating icons by a mouse."[188] It further observed that "Apple cannot get patent-like protection for the idea of a graphical user interface."[189] Numerous courts have followed the *Microsoft* "thin" protection approach to judging the scope of software copyrights, in part to ensure that software copyrights would not be construed to provide patent-like protection.[190]

1995) (analogizing the Lotus command hierarchy to the buttons on a VCR), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

184. *Microsoft*, 799 F. Supp. at 1023. The Ninth Circuit approved of the district court's recognition of "the functional aspects of graphical user interfaces." *Microsoft*, 35 F.3d at 1442 n.10.

185. *Microsoft*, 799 F. Supp. at 1023. Many elements of the Apple GUI had been adopted by other software developers. *Id.* at 1024; *see also Microsoft*, 35 F.3d at 1438 (approving standardization as a ground for limiting the scope of copyright in the Apple GUI).

186. *Microsoft*, 35 F.3d at 1438.

187. *Id.* at 1444.

188. *Id.* Notice the use of both § 102(b) ("process") and merger ("indispensable") terminology. *Id.*

189. *Id.* at 1443. GUI designers face various types of design constraints, including the power and speed of the computer, hardware display operations, and ergonomic factors. *Id.* at 1445.

190. *See, e.g.*, Incredible Techs., Inc. v. Virtual Techs., Inc., 400 F.3d 1007, 1012–14 (7th Cir. 2005) (applying the virtual identity standard of copyright infringement to a golf game interface and concluding that functional features can only be protected by patents); MiTek Holdings, Inc. v. Arce Eng'g Co., 89 F.3d 1548, 1558–59 (11th Cir. 1996) (invoking *Microsoft* and applying the "virtual identicality" standard to software copyright infringement); *see also* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 705 (2d Cir. 1992) (stating that in view of the "highly functional, utilitarian" nature of programs, "it may well be that the Copyright Act serves as a relatively weak barrier against public access to the theoretical interstices behind a program's source and object codes"). By directing courts to filter out an extensive list of program elements before comparing the works at issue to determine if infringement occurred, *Altai*, 982 F.2d at 706–11, *Altai* was a precursor to the thin protection approach to software copyrights endorsed in *Microsoft*.

4. The Merger Approach

A more distinctive judicial strategy for managing copyright and patent boundaries in software relies on the merger doctrine.[191] When an idea or function can, as a practical matter, be expressed in only a very limited number of ways, copyright protection should be withheld from the merged idea/function and its expression.[192] The CAFC's *Atari* decision was the first appellate court opinion to acknowledge the potential applicability of a process/expression merger doctrine in software cases.[193] The CAFC concluded that "if the patentable process [in a program] and its expression are indistinguishable or inextricably intertwined, then 'the process merges with the expression and precludes copyright protection.'"[194] Had Atari copied only so much of the Nintendo data stream as was necessary to achieve compatibility with the then-current version of the Nintendo platform, the CAFC would have ruled in Atari's favor on merger grounds.[195] However, because Atari copied more than was necessary, its merger defense failed.[196] *Atari* would have left the role of protecting such a process to patent law.

Atari's process/expression merger doctrine was influential in *Lexmark International, Inc. v. Static Control Components, Inc.*[197] This lawsuit challenged Static's copying of a small program installed in Lexmark cartridges so that Static's chip customers could make their printer

191. Courts often trace the merger doctrine to *Baker*. *See, e.g.*, Arica Inst., Inc. v. Palmer, 970 F.2d 1067, 1076 (2d Cir. 1992).

192. *See, e.g.*, Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738, 741–42 (9th Cir. 1971) (holding that the expression in a jeweled bee pin was inseparable from its idea).

193. *See* Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 839–40 (Fed. Cir. 1992).

194. PRG-Schultz Int'l, Inc. v. Kirix Corp., No. 03C1867, 2003 WL 22232771, at *4 (N.D. Ill. Sept. 22, 2003) (quoting *Atari*, 975 F.2d at 839–40). Other software cases have recognized this type of merger as well. *See, e.g.*, *id.* (auditing program expression merged with a process); *see also* CONTU REPORT, *supra* note 47, at 50 ("[W]hen specific instructions, *even though previously copyrighted*, are the only and essential means of accomplishing a given task, their later use by another will not amount to an infringement." (emphasis added)). The merger approach has also been used to manage copyright/patent boundaries in non-software cases. *See* Samuelson, *supra* note 2, at 1524–28.

195. *Atari*, 975 F.2d at 839–40; *see also* Harbor Software, Inc. v. Applied Sys., Inc., 925 F. Supp. 1042, 1048–49 (S.D.N.Y. 1996) (finding that process and expression for some nonliteral elements of software had merged); NEC Corp. v. Intel Corp., No. C-84-20799-WPG, 1989 WL 67434, at *15 (N.D. Cal. Feb. 6, 1989) (noting that Intel's patented hardware limited NEC's choices in creating microcode).

196. Professor Cohen disagreed: "The 'surplus' functions in the 10NES program did not become expression by virtue of their surplusage. They were designed to perform particular functions at the interface between the console and cartridge . . . ." Cohen, *supra* note 144, at 1146.

197. 387 F.3d 522, 535 (6th Cir. 2004).

cartridges interoperate with Lexmark printers.[198] The Lexmark cartridge program performed a digital handshake with the Lexmark printer software to authenticate the cartridge so it would work in Lexmark printers.[199] Lexmark claimed that Static's copying infringed copyright. Static defended by challenging the validity of Lexmark's copyright and asserting a fair use defense.[200]

Static appealed the district court's issuance of a preliminary injunction.[201] The U.S. Court of Appeals for the Sixth Circuit characterized the Lexmark cartridge program as a "lock-out" code which "fall[s] on the functional-idea rather than the original-expression side of the copyright line."[202] It recognized that many device makers had employed similar software security systems to thwart interconnections with unlicensed complementary products.[203] The court decided that "[t]o the extent compatibility requires that a particular code sequence be included in the component device to permit its use, the merger and *scènes à faire* doctrines generally preclude the code sequence from obtaining copyright protection."[204]

The Sixth Circuit invoked *Baker* and *Altai*, among other cases, as differentiating the roles of copyright and patent in the protection of functional works such as software.[205] In keeping with *Atari*, the court recognized that patentable processes and program instructions could merge, which would preclude copyright protection for the merged materials.[206]

Lexmark's expert testified that Static could have developed compatible software written somewhat differently from Lexmark's code.[207] However, the Sixth Circuit decided that, as a practical matter, the

198. *Id.* at 529. Lexmark customers could purchase refillable Lexmark cartridges, or, for a lower price, non-refillable cartridges subject to a license requirement enforced by installed software. *Id.* at 529–30. Static's chips undermined this strategy, enabling Lexmark competitors to sell their cartridges to Lexmark customers. *Id.*

199. *Id.* at 530. The Lexmark Toner Loading Program's code consisted of a relatively small number of commands, which varied somewhat for each printer. *Id.* at 529–30. The printer-side software was more substantial. *Id.*

200. *Id.* at 530–32. Lexmark also claimed that Static violated the anti-circumvention rules, 17 U.S.C. § 1201(a)(1)(A), (a)(2) (2012), because Static's chips bypassed the software security code embedded in Lexmark's printers and cartridges. *Lexmark*, 387 F.3d at 546–50. The Sixth Circuit rejected this claim. *Id.*

201. *Id.* at 529.

202. *Id.* at 536.

203. *Id.*

204. *Id.* The court cited approvingly to *Accolade* and *Atari*. *Id.*

205. *Id.* at 534–35.

206. *Id.* at 535.

207. *Id.* at 539.

expressive choices available to Static were very limited.[208] It consequently ruled that insofar as the Lexmark code was necessary to enable non-Lexmark cartridges to work in Lexmark printers, Static's copying of that code did not infringe.[209] The main justification that stands out in *Lexmark* is the merger rationale.[210] In the background, though, as in *Atari*, was a concern that copyright protection for programs not be interpreted to give patent-like protection to functional aspects of programs.

5. The Explanation/Use Distinction Approach

The explanation/use distinction, like several other doctrines of U.S. copyright law,[211] traces back to the Supreme Court's *Baker* decision. The Court cryptically stated that:

> The description of the [useful] art in a book, though entitled to the benefit of copyright, lays no foundation for an exclusive claim to the art itself. The object of the one is explanation; the object of the other is use. The former may be secured by copyright. The latter can be secured, if it can be secured at all, by letters-patent.[212]

The explanation/use distinction was significant in *Baker* because Selden's widow had announced her intent to sue Baker's customers for using the allegedly infringing forms to keep their books once she prevailed in her lawsuit against Baker.[213] Had Selden obtained the patent he once sought, that patent would have given him an exclusive right to control uses of that system, not just making and selling forms embodying his invention.[214]

---

208. *Id.* at 539–41 (explaining that efficiency and functionality precluded material changes to the Lexmark code and that proposed alternatives were trivial and inefficient, not expressive).

209. *Id.* at 542–44.

210. *Id.* at 542 ("[I]f any single byte of the Toner Loading Program is altered, the printer will not function."); *see also id.* at 551 (Merritt, J., concurring) (stating that the Lexmark Toner Loading Program is uncopyrightable under the merger and *scènes à faire* doctrines).

211. At least eight major doctrines of U.S. copyright law owe their origins to *Baker*. *See* Samuelson, *supra* note 22, at 180–92.

212. Baker v. Selden, 101 U.S. 99, 105 (1879); *see* Samuelson, *supra* note 2, at 1529–31 (discussing the Office's refusal to register a claim in a DNA sequence because its elements were selected and arranged to produce a functional result in a biological organism, not for expressive purposes).

213. Samuelson, *supra* note 22, at 167–68.

214. 35 U.S.C. § 271 (2012). Baker's forms were somewhat different from Selden's forms and an improvement over them. *See* Samuelson, *supra* note 22, at 161–62, 193 (stating that the principal difference between the forms was in how they treated accounts as well as how Baker's forms had several key advantages over Selden's, such as their ease of use). Hence, Baker's forms might not have infringed a Selden patent on his system.

The explanation/use distinction has occasionally played a role in the subsequent caselaw.[215] It largely fell out of favor after Mazer's infelicitous reliance on it in his challenge to Stein's copyright in a statuette intended for use as the base of Stein's lamps.[216] Because *Baker* recognized the viability of copyrights in aesthetic creations as well as in explanatory works,[217] Mazer's reliance on *Baker*'s explanation/use distinction was an especially weak part of his argument. Treatise author Melville B. Nimmer's longstanding criticism of this distinction may explain why it has had relatively little currency until quite recently.[218]

The First Circuit's *Borland* decision invoked this distinction when deciding that the Lotus 1-2-3 command hierarchy was an uncopyrightable method of operation.[219] A "means by which a person operates something, whether it be a car, a food processor, or a computer" was within the § 102(b) exclusion.[220] A description of such a method would not confer exclusive rights over "the method of operation itself."[221] And "[i]f specific words are essential to operating something, then they are part of a 'method of operation' and, as such, are unprotectable."[222] In *Borland,* Lotus had chosen the command words and structured them not for explanatory purposes, but rather as a means for accomplishing functional tasks: "[U]sers operate Lotus 1-2-3 by using the Lotus menu command hierarchy" because "the entire Lotus menu command hierarchy is essential to operating Lotus 1-2-3."[223]

---

215. The Seventh Circuit invoked the distinction in *Taylor Instrument Cos. v. Fawley-Brost Co.*, 139 F.2d 98, 100 (7th Cir. 1943).

216. Mazer v. Stein, 347 U.S. 201, 212–17 (1954). That is, Mazer argued that Stein had created the statuette intending to use it as the base of a lamp, not as a work of art. *See infra* text accompanying notes 255–58 (discussing *Mazer*).

217. *Baker*, 101 U.S. at 103–04 (recognizing a copyright in "ornamental designs, or pictorial illustrations addressed to the taste"). The statuette in *Mazer* was ornamental.

218. NIMMER, *supra* note 17, § 37.3. Subsequent versions of the Nimmer treatise remained critical of the explanation/use distinction. *See, e.g.*, 1 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 2.18[D][1] (2015), *reprinted in* 11 NIMMER & NIMMER 2018, *supra* note 17, app. 67C. However, the recently revised version is more receptive to this doctrine. *See* NIMMER & NIMMER 2018, *supra* note 17, § 2A.07[D][2][a][ii].

219. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815–16 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996). For an excellent discussion of *Borland*, see generally Weinreb, *supra* note 14. While concurring with the First Circuit's ruling, Weinreb found its § 102(b) analysis unsatisfactory. *Id.* at 1207–08.

220. *Borland*, 49 F.3d at 815.

221. *Id.* The court was sympathetic to Lotus' customers who had become accustomed to using the Lotus command structure and macro system. *Id.* at 817–18.

222. *Id.* at 816.

223. *Id.* David Hayes contends that courts should differentiate the explanation/use distinction and a copying-for-use/copying-for-explanation distinction. He considers the former a subject matter eligibility doctrine, while the latter is relevant to infringement analysis (i.e., copying the Lotus command hierarchy for use purposes, as Borland did, would not infringe, but copying it for

The First Circuit quoted *Baker* on the explanation/use distinction in support of its method of operation analysis.[224] Judge Michael Boudin's concurrence characterized the Lotus program as "a *means* for causing something to happen; it has a mechanical utility, an instrumental role, in accomplishing the world's work."[225] To enforce Lotus' copyright against Borland would, he said, have "some of the consequences of *patent* protection in limiting other people's ability to perform a task in the most efficient manner."[226]

The explanation/use distinction has recently garnered renewed support from courts and copyright scholars.[227] Professor Wendy Gordon, has criticized the CAFC's *Oracle* decision for failing to appreciate the utility of the Java API declarations as tools for creating new works of authorship.[228] Professor Gordon argues that "a copyright owner should have no prima facie rights over copying behavior where (1) the goals of the copying are 'use' (behavior in the realm of utility patent) and (2) the copying is done solely for goals unrelated to the expressiveness of the plaintiff's work of authorship."[229]

Gordon perceives the explanation/use distinction as "implement[ing] the deference that, pursuant to Congressional command and Supreme Court precedent, U.S. copyright law must give to patent law."[230] This distinction "operates by limiting the *scope* of the exclusive rights a

---

explanatory purposes would more likely infringe). Telephone Interview with David L. Hayes, Partner, Fenwick & West LLP (Nov. 9, 2017).

224. *Borland*, 49 F.3d at 816–17 (quoting Baker v. Selden, 101 U.S. 99, 104–05 (1879)).

225. *Id.* at 819 (Boudin, J., concurring).

226. *Id.*

227. *See* Bikram's Yoga Coll. of India, L.P. v. Evolation Yoga, LLC, 803 F.3d 1032, 1040–41 (9th Cir. 2015) (invoking the explanation/use distinction); *see also* ABRAHAM DRASSINOWER, WHAT'S WRONG WITH COPYING? 13 (2015) ("[In *Baker*,] [t]he defendant used the forms as a tool but not *as a work*, and was therefore not liable in copyright. . . . *Baker* thus turns on a crucial distinction between the work as a communicative act and its material form as its physical embodiment. Use of the physical embodiment for noncommunicative purposes does not give rise to liability."); *see also* Cohen, *supra* note 144, at 1145 (discussing interoperability-related routines as methods of operation relying on the use/expression distinction).

228. For this criticism, see generally Wendy J. Gordon, *How* Oracle *Erred: The Use/Explanation Distinction and the Future of Computer Copyright*, *in* COPYRIGHT LAW IN AN AGE OF LIMITATIONS AND EXCEPTIONS 375 (Ruth Okediji ed., 2017).

229. *Id.* at 376. The explanation/use distinction resembles the rule that a copyright in a drawing of a useful article does not confer copyright protection over the useful article depicted therein. *See, e.g.*, Fulmer v. United States, 103 F. Supp. 1021, 1021–22 (Ct. Cl. 1952) (holding that a copyright in plaintiff's drawing did not give it exclusive rights to control the manufacture of parachutes embodying that design; invoking *Baker*, such exclusive rights can only be obtained through a patent). This rule is now codified at 17 U.S.C. § 113(b) (2012).

230. Gordon, *supra* note 228, at 376; *see also* McKenna & Sprigman, *supra* note 30, at 543 (discussing the notion of patent supremacy over other forms of IP).

copyright owner might otherwise possess, not by targeting the copyrightability of what plaintiff produced."[231]

Gordon offers this non-software example to illustrate the distinction:

> [I]t is an expressive use when the publisher of a how-to book on home repair copies someone else's copyrighted passage explaining how to rewire a lamp instead of writing his own instructions. It is a nonexpressive use when a homeowner applies the same copyrighted passage to the task of actually rewiring lighting fixtures. Copying text to convey an explanation or to serve other expressive goals belongs to the realm of copyright; copying to build a functional invention instead belongs to the realm of patent.[232]

In Gordon's view, Google made non-expressive uses of the Java method headers to enable programmers trained in the Java programming language to create new programs or adapt old ones to run on the Android platform.[233] The declarations were tools for creating programs because they identified particular functions to be performed, just as the Lotus command hierarchy provided tools for user-constructed macros.[234]

Gordon might, although she did not, have added that many books set forth the Java API, including many thousands of its declarations, in order to explain these functions and how to use the declarations to create programs.[235] These books arguably publish elements of the Java API for explanatory purposes. Oracle has not objected to these uses of the Java API, but only Google's use of them as tools for its Android platform.[236]

C. *The Scope of Software Copyrights in the Post-*Altai *Caselaw*

Each of the judicial strategies for avoiding overlap of copyright and patent protections in software has merit within its doctrinal framing. The "thin" protection strategy, with its endorsement of filtration of numerous categories of software elements and of a virtual identity standard to support an infringement finding, may be the surest way for courts to

231. Gordon, *supra* note 228, at 376.

232. *Id.* at 380.

233. *Id.* at 381–82.

234. *Id.* at 427. Gordon did not address whether she believed the Java declarations were patentable.

235. *See, e.g.*, Oracle Am., Inc. v. Google Inc., No. C 10-03561 WHA, 2016 WL 3181206, at *5 (N.D. Cal. June 8, 2016) (mentioning books setting forth and explaining the Java API), *rev'd*, 886 F.3d 1179 (Fed. Cir. 2018).

236. The command hierarchies in *Borland* were readily visible to users of the software. The command structure of the Java declarations was not visible to users of Android. Programmers learn Java commands and their structures by studying the Java Standard Edition text that sets forth the Java API or reading books that teach programmers how to use this API. *Id.*

ensure that copyright is not being used to confer patent-like protection to software innovations.[237] Yet, courts also averted overlap problems when relying on the conception of copyright and patent protecting different aspects of programs, on the § 102(b) exclusions, and on the merger doctrine. The cases have achieved comparable outcomes while contributing to a more refined jurisprudence about copyright-protectable and unprotectable structures.

Considering these developments, it can be said with reasonable assurance that concerns about the overbroad scope of software copyrights expressed in the late 1980s and 1990s subsided considerably once the *Altai* decision garnered progeny. The software copyright caselaw since then has generally followed the lead of *Altai* and its progeny in engaging in rigorous filtration of unprotectable elements and focusing infringement analysis on software design elements that have some expressive character.[238]

Yet, the software copyright cases since 2000, although generally following *Altai* and other similar cases, rarely mention the potential patentability of software innovations when judging copyright infringement claims. Perhaps as a consequence, the issue of whether copyright and patent can provide overlapping protections or are mutually exclusive has rarely surfaced in the post-*Altai* cases. However, in the few cases to which this Article now turns, it has.

---

237. The Ninth Circuit in *Microsoft* regarded thin protection as the necessary result of applying numerous limiting doctrines to those parts of the Apple GUI claimed to infringe. *See supra* Section II.B.3.

238. Most software copyright cases since *Altai* have engaged in highly dissective analyses of software copyright claims. *See, e.g.*, R.C. Olmstead, Inc. v. CU Interface, LLC, 606 F.3d 262, 275–76 (6th Cir. 2010) (affirming summary judgment for defendant because plaintiff failed to specify which elements of its credit union processing software were original expressions and to filter out unprotectable elements); Real View, LLC v. 20-20 Techs., Inc., 683 F. Supp. 2d 147, 157, 162 (D. Mass. 2010) (finding that screen displays and GUI were not infringed because too many elements were unprotectable under § 102(b) and the doctrines of merger and *scènes à faire*); Comput. Access Tech. Corp. v. Catalyst Enters., Inc., No. C-00-4852-DLJ, 2001 WL 34118030, at *14–17 (N.D. Cal. June 13, 2001) (holding that many aspects of the plaintiff's GUI were unprotectable as industry standards, efficient designs, or as dictated by function; sufficient differences in the GUIs precluded the relief that CATC sought).

There have been, however, a few problematic post-*Altai* cases ruling that the defendants' use of the same methods infringed. *See, e.g.*, Aspen Tech., Inc. v. M3 Tech., Inc., 569 F. App'x 259, 270 (5th Cir. 2014) (finding infringement based on the use of the same steps, parsing methods, and power function device in chemical refinery programs); *see also* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1356–57 (Fed. Cir. 2014) (opining that copyright protection is available to protect program methods). Some courts have extended protection to command languages. *See, e.g.*, McEnroe v. Mantissa Corp., No. 14-cv-12320-LTS, 2016 WL 7799636, at *13 (D. Mass. Feb. 29, 2016) (holding that a command language was copyrightable).

III. THE CAFC'S ERRONEOUS ENDORSEMENT OF COPYRIGHT/PATENT OVERLAPS IN *ORACLE*

The CAFC's *Oracle* decision is the only precedent to have endorsed overlapping copyright and patent protections for nonliteral elements of software such as APIs.[239] The issue came up in two other cases, but appellate courts had different views. *Oracle* was deeply flawed in its treatment of the overlap issue and its endorsement of overlap should be given no weight in future cases. Courts should be more receptive to well-drawn arguments that the patentability of some nonliteral elements of programs is evidence that copyright protection should be withheld.

A. *Reversals of Copyright/Patent Overlap-Approving Decisions in Software Cases*

Except for *Oracle*, courts have generally been unreceptive to arguments that copyright and patent can provide overlapping protections for the same aspects of software. Two lower court decisions accepting this possibility were reversed on appeal.

One was the district court decision in *Gates Rubber Co. v. Bando America, Inc.*[240] Bando argued that its use of some of the same algorithms as the Gates program was non-infringing because algorithms were patent, not copyright, subject matter.[241] The trial court rejected this argument, saying that "[s]uch a holding would tend to fragment further the rather tenuous continuity found in copyright law concerning computer programs."[242]

The Tenth Circuit reversed, stating that program processes, such as algorithms, were unprotectable by copyright law under § 102(b), even

239. Two litigants unsuccessfully argued that computer programs were uncopyrightable because they were patentable. *See* Aharonian v. Gonzales, No. C 04-5190 MHP, 2006 WL 13067, at *7 (N.D. Cal. Jan. 3, 2006) (denying a declaratory judgment that computer programs could not be copyrighted because they consisted of patentable algorithms and data structures); Apple Comput., Inc. v. Franklin Comput. Corp., 545 F. Supp. 812, 8125 (E.D. Pa. 1982) (denying a motion for a preliminary injunction because copyright law only protects communicative authorship, not utilitarian works such as software, the latter being protectable only by patents), *rev'd*, 714 F.2d 1240, 1250–53 (3d Cir. 1983) (ordering a preliminary injunction because the exact copying of Apple operating system programs was copyright infringement).

240. 798 F. Supp. 1499 (D. Colo. 1992), *aff'd in part, vacated in part sub nom*. Gates Rubber Co. v. Bando Chem. Indus. Ltd., 9 F.3d 823 (10th Cir. 1993). Gates' program enabled its sales staff to input certain variables and perform calculations needed to assess which of Gates' industrial belts would work best for customers' machines. *Id.* at 1503. Bando developed a program that performed the same types of calculations. *Id.* Gates charged nonliteral infringement. *Id.* at 1518.

241. *Id.*

242. *Id.*

though they might be eligible for patenting.[243] It recognized that programs designed to perform complex calculations would necessarily embody numerous methods and processes that copyright could not protect.[244] The exclusion of processes from the scope of copyright in software was "one of the most important" of the utilitarian elements Congress meant to make unprotectable by copyright.[245] The court quoted from the legislative history:

> Some concern has been expressed lest copyright in computer programs should extend protection to the methodology or processes adopted by the programmer, rather than merely to the "writing" expressing his ideas. Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law.[246]

Consistent with *Altai*, the Tenth Circuit in *Gates* recognized that such processes should be filtered out before assessing the infringement claim.[247]

A second copyright/patent overlap-approving decision was the district court's first opinion in *Borland*.[248] The court observed that "[t]he mere fact that patent law allows a means of legal protection for a process . . . does not establish that there is not also some protection in copyright . . . ."[249] In its view, copyright and patent provided rights that were perhaps "not coextensive, but it is equally clear that there is no particular reason to believe there should never be any area of overlap."[250] The court cited to the copyright/(design)patent overlap-endorsing dicta in

243. *See Gates Rubber Co.*, 9 F.3d at 836–37. The Tenth Circuit reaffirmed the exclusivity of copyright and patent protections in protecting software innovations. *Id.*

244. *Id.* at 830, 836. The district court's infringement ruling was based on similarities in mathematical constants, algorithms, menus, sorting criteria, fundamental tasks, install files, and overall data and control flows. *Id.* at 842.

245. *Id.* at 836.

246. *Id.* at 836–37 (quoting H.R. REP. NO. 94-1476, at 57 (1976)). This is the language from the legislative history that the *Oracle* decision neglected to cite. *See infra* note 269.

247. *Gates Rubber Co.*, 9 F.3d. at 842, 845. The Tenth Circuit remanded the case for clarification of Gates' copyright claims based on similarities in menus, sorting criteria, data flow and control flow. *Id.* at 844–46.

248. For this decision, see generally Lotus Dev. Corp. v. Borland Int'l, Inc. 788 F. Supp. 78 (D. Mass. 1992), *rev'd*, 49 F.3d 807 (1st Cir. 1995).

249. *Id.* at 91.

250. *Id.*

Mazer, asserting that this "precedent recognizes some overlap."[251] It perceived no reason why patent and copyright protection could not be available for such things as program command structures.[252]

Although the First Circuit did not explicitly repudiate the district court's dicta on copyright/patent overlaps, it held that Borland's copying of the Lotus command hierarchy did not infringe copyright.[253] In view of its invocation of § 102(b)'s method of operation exclusion, that court's decision is more consistent with the many other precedents that have sought to maintain separate domains for copyright and patent in the protection of program innovations than with the district court's overlap endorsement.[254]

B. *The CAFC's Flawed Acceptance of a Software Copyright/Patent Overlap in* Oracle

To support its view that copyright and patent could provide overlapping protections for software APIs, the *Oracle* court invoked the Supreme Court's rejection of an IP exclusivity argument in *Mazer v. Stein*.[255] Mazer had argued that Stein's claim of copyright in a statuette of a Balinese dancer was invalid because Stein should have obtained a design patent for exclusive rights to manufacture the statuette as an ornamental design for an article of manufacture (that is, as the base of a lamp).[256] Under Mazer's conception of *Baker*, the unpatented lamp base design was in the public domain and available for unrestricted copying.[257] The Court disagreed, observing that "[n]either the Copyright Statute nor any other says that because a thing is patentable it may not be copyrighted."[258] This is the dictum with which the CAFC fended off Google's mutual exclusivity argument in *Oracle*.[259] This was the only judicial precedent to which the CAFC adverted in that part of its opinion.[260]

The CAFC did not address Google's patent-not-copyright-subject-matter defense until the very end of its opinion, characterizing it as a

---

251. *Id.* (citing Mazer v. Stein, 347 U.S. 201, 217 (1954)); *see infra* text accompanying notes 266–68 (explaining why *Mazer* does not support the district court's assertion).

252. *Borland*, 788 F. Supp. at 91. The court did not opine whether it thought the Lotus command structure and macro system were both patentable and copyrightable.

253. *Borland*, 49 F.3d at 818; *see supra* text accompanying note 228.

254. *See supra* text accompanying notes 153–56, 219–26 (discussing *Borland*).

255. *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1380 (Fed. Cir. 2014) (quoting *Mazer*, 347 U.S. at 217).

256. *See Mazer*, 347 U.S. at 215–16.

257. *Id.* at 217.

258. *Id.*

259. *Oracle*, 750 F.3d at 1380.

260. *See id.* at 1377–81.

"policy-based argument" that attacked the very existence of copyright protection for programs, as if Google was claiming that only patents could protect program innovations.[261] Until Congress or the Supreme Court repudiated software copyrights, the CAFC felt dutybound to uphold the copyrightability of software.[262]

The CAFC's response to Google's exclusivity argument is unpersuasive for at least four reasons. First, the court misconstrued Google's argument. Google was not contending that original programs could not be protected by copyright law and were only protectable by patent law.[263] Such an argument would have been specious, and as a major developer of software itself, Google had every reason to want original software to be copyrightable. Rather, Google's argument was that the scope of the valid copyright in the Java Standard Edition document, which sets forth elements of the Java API, did not extend to the declarations it used in Android because they were the names of functions that programmers needed to use to write programs in Java.[264]

Second and most significantly, the CAFC failed to recognize that the Supreme Court in *Mazer* actually supported Google's exclusivity defense. The dictum quoted by the CAFC arose in the context of a real, albeit partial, overlap in the subject matters protectable by copyright and design patent law.[265] Stein's statuette in *Mazer* qualified as a work of art under U.S. copyright law, although when used as the base of a lamp, the statuette was also eligible for design patent protection as an ornamental design of an article of manufacture.[266] The CAFC overlooked the unequivocal statement in *Mazer* that "the Mechanical Patent Law and Copyright Laws are *mutually exclusive*,"[267] as well as *Mazer's* approving citation to *Taylor* and three other *Baker* progeny, all of which upheld exclusivity of copyright and patent protections.[268]

---

261. *See id.* at 1379–81. The CAFC seemingly thought that Google was making the same flawed arguments as *Franklin* and *Aharonian* had done in earlier cases. *See supra* text accompanying note 239.

262. *See Oracle*, 750 F.3d at 1381.

263. *See* Brief of Appellee & Cross-Appellant Google Inc. at 29–33, *Oracle*, 750 F.3d 1339 (Nos. 2013-1021, -1022) [hereinafter Google Brief].

264. *Id.* at 15. Google relied more heavily on its § 102(b) defense than the copyright/patent exclusivity defense. *Id.* at 31–40, 57–65.

265. *See Oracle*, 750 F.3d at 1380.

266. *See* Mazer v. Stein, 347 U.S. 201, 215–17 (1954). Well before *Mazer*, the Court had recognized that design patents and copyrights had overlapping subject matters. *See, e.g.*, DeJonge & Co. v. Breuker & Kessler Co., 235 U.S. 33, 36 (1914) (noting the lower court's opinion that a pictorial design was eligible for copyright and a design patent).

267. *Mazer*, 347 U.S. at 215 n.33 (emphasis added).

268. *See id.* at 217 n.39 (citing additional *Baker* progeny); *see also* NIMMER & NIMMER 2018, *supra* note 17, § 2A.07[D][4] (characterizing the CAFC's *Oracle* opinion as a "misconstru[al of] *Mazer*").

Third, the CAFC's recitation of the legislative history of § 102(b) omitted reference to the most relevant part, which expressed the intent for § 102(b) to limit the scope of software copyrights: "[s]ection 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law."[269] This was supposed to allay concerns that software copyrights would be construed to extend protections to methods and processes. By its earlier statements endorsing copyright protection for methods,[270] the CAFC's decision is directly contrary to the text of § 102(b) and the pro-competitive policies it was intended to embody.

Fourth, the CAFC ignored that its endorsement of the copyright/patent overlap in *Oracle* was inconsistent with several appellate court decisions in software IP cases,[271] including its *Atari* decision, which averred that "patent and copyright laws protect *distinct aspects* of a computer program."[272] Although the *Oracle* court cited approvingly to that decision,[273] it did not explain why it was deviating from its previous "distinct aspects" perspective.

The overwhelming majority of scholarly commentary on the CAFC's *Oracle* decision has, moreover, strongly criticized it,[274] including its

---

269. H.R. REP. NO. 94-1476, at 57 (1976).

270. *See Oracle*, 750 F.3d at 1356–57.

271. *See supra* Section II.B. The Ninth Circuit interpretation of § 102(b) in *Bikram's Yoga Coll. of India, L.P. v. Evolation Yoga, LLC*, 803 F.3d 1032 (9th Cir. 2015), further undermines the *Oracle* decision. *See, e.g.*, Brief of Amicus Curiae Elec. Frontier Found. in Support of Defendant-Appellee at 4–6, Cisco Sys., Inc. v. Arista Networks, Inc., No. 2017-2145 (Fed. Cir. Dec. 26, 2017) (explaining why *Bikram Yoga* calls into question the copyrightability analysis in *Oracle*).

272. Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 839 (Fed. Cir. 1992) (emphasis added) (citing Baker v. Selden, 101 U.S. 99, 103 (1879)); *see also* Hutchins v. Zoll Med. Corp., 492 F.3d 1377, 1384 (Fed. Cir. 2007) (citing Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995)) (upholding a trial court ruling against plaintiff's copyright claim on the ground that the technological process embodied in the defendant's program was not protectable expression).

273. *See Oracle*, 750 F.3d at 1357.

274. *See, e.g.*, Armstrong, *supra* note 17, at 154–55; Clark D. Asay, *Copyright's Technological Interdependencies*, 18 STAN. TECH. L. REV. 189, 235 (2015); Gordon, *supra* note 228, at 326, 329–30; Joseph Gratz & Mark A. Lemley, *Platforms and Interoperability in* Oracle v. Google, 31 HARV. J.L. & TECH. 603, 605–06 (2018); Sean Hogle, *Software Copyright and Innovation after* Oracle v. Google, 40 NEW MATTER 1, 2 (2015); Peter S. Menell, *Rise of the API Copyright Dead?: An Updated Epitaph for Copyright Protection of Network and Functional Features of Computer Software*, 31 HARV. J.L. & TECH. 305, 307 (2018); Samuelson, *supra* note 8, at 1256; Pamela Samuelson, *Three Fundamental Flaws in CAFC's* Oracle v. Google *Decision*, 37 EUR. INTELL. PROP. REV. 702, 702–08 (2015); Fred von Lohmann, *The New Wave: Copyright and Software Interfaces in the Wake of* Oracle v. Google, 31 HARV. J.L. & TECH. 517, 519 (2018);

endorsement of overlapping copyright and patent protection for software innovations.[275] Although the Nimmer treatise once offered some support for *Oracle*'s copyright/patent overlap ruling, the latest version of the treatise now criticizes this aspect of the CAFC's ruling.[276]

The weaknesses of the CAFC *Oracle* decision's treatment of the copyright/patent overlap issue and the *Mazer-Baker* exclusivity statements notwithstanding, the Supreme Court has sometimes been unreceptive to categorical exclusivity arguments in IP cases.[277] Some IP scholars and lawyers, moreover, have hesitated to endorse a categorical exclusivity approach to managing the copyright/patent boundary as to software innovations.[278] Although many strongly support exclusivity and regard the patentability of nonliteral elements of software as a reason to construe copyright as providing a narrow scope of protection,[279] others view copyright and patent as providing software developers with a choice about which type of IP protection to pursue as a business strategy.[280]

Jonathan Band, The Protectability of Application Program Interfaces: Oracle America v. Google 1–2 (Aug. 14, 2015) (unpublished manuscript), https://papers.ssrn.com/sol3/papers.cfm?Abstract_id=2186628 [https://perma.cc/QYJ8-BGNZ]. *But see* Jonathan Ambrose, Oracle America, Inc. v. Google, Inc.*: The Only Nonliteral Aspects of Java APIs Protected Under Copyright Law Are the Ones Nobody Wants to Copy*, 14 N.C. J.L. & TECH. ON. 1, 5 (2012) (endorsing the district court's copyrightability ruling).

275. *See* Samuelson, *supra* note 8, at 1289–91.

276. *Compare* NIMMER & NIMMER 2017, *supra* note 17, § 2A.07[A], *with* NIMMER & NIMMER 2018, *supra* note 17, § 2A.07[D][4]. The Goldstein treatise obliquely criticizes the CAFC's *Oracle* decision. *See* 1 PAUL GOLDSTEIN, GOLDSTEIN ON COPYRIGHT § 2.15.2.1, at 2:198 n.87.1 (3d ed. 2005 & 2015 Supp.) (stating that CAFC in *Oracle* "purportedly" followed Ninth Circuit precedents).

277. *See, e.g.*, J.E.M. Ag Supply, Inc. v. Pioneer Hi-Bred Int'l, Inc., 534 U.S. 124, 127 (2001) (rejecting the argument that novel plants were unpatentable because Congress intended for them to be protected only under the Plant Variety Protection Act); TrafFix Devices, Inc. v. Mktg. Displays, Inc., 532 U.S. 23, 34–35 (2001) (declining to address whether a product configuration would be categorically ineligible for trade dress protection if patented); Mazer v. Stein, 347 U.S. 201, 217 (1954) (rejecting the argument that a lamp base statuette could be protected only by design patent law). The Court was unconvinced by Borland's categorical exclusivity argument as well. *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 516 U.S. 233, 233 (1996) (affirming the First Circuit's ruling by a 4-4 vote). Borland's brief to the Supreme Court relied heavily on its categorical exclusivity argument. *See* Borland Brief, *supra* note 155, at 22–44.

278. *See* Chisum et al., *supra* note 18, at 17–18 (stating that there is no consensus among IP scholars regarding whether the patentability of program innovations is relevant to the scope of copyright protection for SSO). *See generally* Samuelson, *supra* note 18 (reporting on a survey of IP lawyers about copyright and patent protection for specific aspects of programs).

279. *See, e.g.*, Karjala, *Relative Roles*, *supra* note 17, at 49–50; Lundberg et al., *supra* note 16, at 252; *see also supra* note 17 (listing more sources that take this view).

280. *See, e.g.*, House Hearings, *supra* note 107, at 95 (testimony of Pamela Samuelson) (reporting that IP lawyers would characterize nonliteral software structures as methods when they wanted to patent them and as SSO when asserting copyright); Samuelson, *supra* note 18, at 270–71.

That being said, no court has ever actually held that any nonliteral element of software was both copyright and patent protectable.[281] Nor has any judicial opinion endorsed the notion that software developers could claim copyright protection for nonliteral elements of software for which utility patents could have issued but had not been obtained, or as to which a utility patent had issued, but then expired. Nor has any commentator forthrightly asserted that copyright should serve as a gap-filler for software designs that fall within the patent domain, but whose creator failed either to seek or to get a patent.

Yet, the potential for copyright and patent to provide overlapping or substitutive protection for program innovations is real, in no small part due to the lack of clarity in both the copyright and patent caselaw about exactly which elements of software fall within the scope of each law.[282] As Judge Boudin observed in *Borland*, some aspects of programs, such as user interface command structures that are visible to users, "look hauntingly like the familiar stuff of copyright," although "the 'substance' probably has more to do with problems presented in patent law."[283] There may, however, be a more nuanced way to raise exclusivity defenses in software copyright cases.

C. *The Taking-Patents-Into-Account Approach*

Google may have made a strategic mistake when raising its copyright/patent exclusivity defense. Instead of arguing that the existence of some patents on command structures or APIs was dispositive evidence that these types of nonliteral elements of software were patent and not copyright subject matter, Google should have argued that such patents are strong evidence that command structures or APIs were unprotectable under § 102(b).[284] This kind of taking-patents-into-account approach[285]

281. Nor has any court ever upheld overlapping copyright and patent protection for the same aspects of non-software creations. *See* Samuelson, *supra* note 2, at 1537.

282. *See, e.g.*, McKenna & Sprigman, *supra* note 30, at 543–44 (expressing concern about the indistinctness of patent subject matter boundaries as an impediment to effective channeling rules for other IP regimes). Part IV of the McKenna and Sprigman article discusses how the copyright/patent boundary ambiguity might be reduced. *Id.* at 540–45.

283. Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 820 (1st Cir. 1995) (Boudin, J., concurring), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

284. The PTO might have issued those patents in error. Or the patents may have been granted at a different level of abstraction than the command structures or APIs at issue in the *Borland* and *Oracle* cases.

285. This taking-patents-into-account has been a viable strategy in some non-software copyright cases. *See, e.g.*, Samuelson, *supra* note 2, at 1531–33.

is consistent with the Supreme Court's decision in *TrafFix Devices, Inc. v. Marketing Displays, Inc.*,[286] as well as with *Baker*[287] and *Taylor*.[288]

TrafFix is, of course, a trade dress, not a copyright, case, but the principle enunciated in that case has application beyond the trade dress domain.[289] The Court in *TrafFix* did not regard the existence of an expired patent as conclusive evidence that the design could not be protected as trade dress.[290] But an expired patent's description of a design's functionality, such as MDI's dual-spring design for roadside signs, was strong evidence that the design was too functional to be protectable trade dress.[291] Because functionality can disqualify designs from copyright as well as trade dress protections,[292] courts should examine software-related patents for descriptions of the patented design's functional advantages that may disqualify it from copyright protection.[293]

The exclusivity argument in *TrafFix* was particularly strong because MDI had actually patented the very same dual-spring design in which it was claiming trade dress rights.[294] That design's functionality was plainly described in that patent, and the patent had expired.[295] It took some chutzpah for MDI to contend that this dual-spring design was non-functional in hopes of extending MDI's exclusive control over the design.[296] Allowing MDI to claim trade dress protection would defeat the well-established patent policy that when a patent has expired, the design is in the public domain and available for unrestricted copying.[297]

---

286. 532 U.S. 23, 30–31 (2001).

287. Baker v. Selden, 101 U.S. 99, 102–03 (1880) (commenting that useful arts that could not be protected by copyright might be protected by patent).

288. Taylor Instrument Cos. v. Fawley-Brost Co., 139 F.2d 98, 99–101 (7th Cir. 1943) (concluding that there is no overlap between copyright and patent protection and observing that though material is no longer patented, it cannot be protected by copyright).

289. *See, e.g.*, Viva R. Moffat, *The Copyright/Patent Boundary*, 48 U. RICH. L. REV. 611, 652–56 (2014); Samuelson, *supra* note 8, at 1292.

290. *TrafFix*, 532 U.S. at 35.

291. *Id.* at 29–30.

292. *See, e.g.*, Nat'l Med. Care, Inc. v. Espiritu, 284 F. Supp. 2d 424, 437 (S.D.W. Va. 2003) (holding that a cabinet for a dialysis center was too utilitarian to be protectable by copyright law).

293. *See, e.g.*, OddzOn Prods., Inc. v. Oman, 924 F.2d 346, 348 (D.C. Cir. 1991) (upholding the denial of copyright registration to a patented KOOSH ball on the grounds that it was a useful article that lacked separable expression). *OddzOn* is discussed in Samuelson, *supra* note 2, at 1524–26.

294. *TrafFix*, 532 U.S. at 29–30. The Solicitor General supported TrafFix's appeal. *Id.* at 25. John G. Roberts, Jr., now Chief Justice of the Supreme Court, was TrafFix's lawyer. *Id.*

295. *Id.* at 25–26.

296. The Sixth Circuit held that MDI's design was eligible for trade dress protection because it was not dictated by function. Mktg. Displays, Inc. v. TrafFix Devices, Inc., 200 F.3d 929, 942 (6th Cir. 1999), *rev'd*, 532 U.S. 23 (2001).

297. *See, e.g.*, Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141, 151 (1989) (holding that Florida's anti-plug mold statute was preempted by federal patent policy).

The *Oracle*[298] case presented less compelling facts for a copyright/patent exclusivity defense than did *TrafFix*.[299] In *Oracle*, the district court mentioned patents on program interfaces, including some owned by Oracle, as suggestive that APIs were patent, not copyright subject matter.[300] But the court did not closely analyze those patents or compare their claims with the Java API declarations in which Oracle claimed copyright.

While the taking-patents-into-account approach has not yet succeeded, that does not mean that the approach can never be successful. When raised in the future, it should be applied with more analytic rigor and attention to detail. With the right set of facts (for instance, a patent on a data structure that described its functional advantages over the state of the prior art), this approach may meet with more success when a copyright claim is made for the same or a very similar software design feature.[301] Because tens, if not hundreds, of thousands of patents have been issued for nonliteral elements of programs in the past three decades,[302] the risk of copyright/patent overlaps as to nonliteral elements of software is far from hypothetical.

IV. RESTORING RIGOROUS FILTRATION TO AVOID CONFERRING PATENT-LIKE PROTECTION TO SOFTWARE THROUGH COPYRIGHT LAW

For more than two decades after *Altai*, the state of software IP law was quite stable in both the copyright and patent domains. Copyright protection for software was understood to be "thin" because courts recognized that programs embodied a relatively high quantum of unprotectable elements that had to be filtered out before assessing

298. 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014).

299. The exclusivity defense was also weak in *Franklin*. Apple Comput., Inc. v. Franklin Comput. Corp., 545 F. Supp. 812 (E.D. Pa. 1982), *rev'd*, 714 F.2d 1240 (3d Cir. 1983). The district court characterized a ROM chip as firmware, *id.* at 824, when the ROM was merely a medium in which the Apple programs were stored. *Id.* at 819–20. It cited *Diamond v. Bradley*, 450 U.S. 381 (1981), as having upheld the patentability of firmware. *Franklin*, 545 F. Supp. at 824. However, *Bradley* merely affirmed the patentability of a data structure (and then only by an equally divided vote). 450 U.S. at 381.

300. *Oracle*, 872 F. Supp. 2d at 996.

301. Software patent cases, by contrast, almost never refer to software copyrights. A rare exception was the Supreme Court's decision in *Gottschalk v. Benson*, 409 U.S. 63, 72 (1972) (quoting from a Presidential Commission report opposing patent protection for software in part because of the availability of copyright protection). *See supra* notes 54–56 and accompanying text.

302. *See, e.g.*, Armstrong, *supra* note 17, at 159–60 (reporting the issuance of 16,000 software patents in 2004, 25,000 in 2009, and nearly 55,000 in 2014).

copyright infringement claims, and patents were understood to be readily available for a wide array of software methods and systems.[303]

In 2014, two important developments—one on the copyright side and one on the patent side—upset this equilibrium. The copyright-side development was the CAFC's *Oracle* decision, which upheld a claim of copyright in parts of the Java API.[304] Although purporting to follow *Altai*,[305] *Oracle* was much closer to *Whelan* in its conception of copyright's scope.[306] It construed § 102(b) as excluding from copyright's scope only abstract ideas and merged elements; it endorsed copyright protection for program methods; it upheld the copyrightability of the Java API because Sun/Oracle had made creative choices in designing it; and it accepted copyright/patent overlaps for nonliteral elements of software, such as APIs and command structures.[307] Like *Whelan* before it, the overbroad software copyright ruling in *Oracle* has emboldened several incumbent firms to initiate similar SSO lawsuits against upstart developers of compatible products.[308] There is a risk that courts after *Oracle* will, even if inadvertently, extend patent-like protection to software through copyright law. Because software developers can easily add a patent claim to their copyright complaints, they can take advantage of the CAFC's exclusive jurisdiction, even if the patent claim falls out of the case at an early stage.

The patent-side development was the Supreme Court's decision in *Alice Corp. v. CLS Bank International*,[309] which struck down a patent for a software-implemented method and system of settling financial transaction risks. *Alice* significantly limited the extent to which software

303. *See generally* Samuelson, *supra* note 8, at 1243 ("[T]he scope of copyright protection in computer programs is generally much thinner than the scope of copyright in conventional literary works . . . because programs embody many functional design elements that lie outside the scope of copyright protection under § 102(b).").

304. Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1381 (Fed. Cir. 2014).

305. *Id.* at 1355–58.

306. *Oracle* gave a very narrow interpretation to § 102(b) exclusions, even opining that program methods might be copyrightable. *Id.* at 1356–57. The similarity between *Whelan* and *Oracle* is discussed in Armstrong, *supra* note 17, at 152–56, and Samuelson, *supra* note 8, at 1222, 1237–67.

307. *Oracle*, 750 F.3d at 1359–72, 1379–81. The CAFC takes a comparably narrow view of functionality as a disqualifier from design patent protection. *See, e.g.*, Ethicon Endo-Surgery, Inc. v. Covidien, Inc., 796 F.3d 1312, 1329–30 (Fed. Cir. 2015) (commenting that determining whether a design is primarily functional or ornamental requires looking at the design as a whole). Only if a design is dictated by function does the CAFC consider it to be too functional to be eligible for patenting. *Id.*

308. *See generally* von Lohmann, *supra* note 274, at 519–27 (discussing cases since *Oracle*).

309. 573 U.S. 208 (2014).

innovations can qualify for patenting.[310] By re-endorsing *Benson* and *Flook,* and emphasizing the abstractness of many software innovations,[311] the Court cast a new shadow on the tens of thousands of software patents issued in the heyday of the *Alappat* and *State Street* everything-under-the-sun-is-patentable decisions. Indeed, many software patents have been invalidated since *Alice*.[312]

A. Oracle *and* Alice *Have Created Perverse Incentives for Developers to Use Copyright to Get Patent-Like Protection for Nonliteral Elements of Programs*

Even before *Oracle* and *Alice*, there were several reasons why it was tempting for software developers to assert copyright protection for nonliteral elements of programs that were functional enough to be patentable, but that remained unpatented. Copyright has several advantages over patents for nonliteral elements of software. First, copyright provides automatic protection, rendering unnecessary the high expenditures of time, money, and effort required to obtain patent protection.[313] Second, copyright requires only a modest amount of originality to qualify for protection, not novelty and nonobviousness, as patent does; and the Copyright Office examination is much less rigorous than the PTO's.[314] Third, with copyright, protection lasts much longer and is much less vulnerable to invalidation than with patents.[315] Fourth, copyright law does not require disclosure of one's creation or specific claiming to acquire the right as patent law does.[316] An SSO copyright claim would bypass patent law's disclosure and claiming requirements. Fifth, the copyright standard for infringement, which assesses the substantial similarity in expression (which courts may not require plaintiffs to define) of the works at issue, is much less precise and demanding than the element-by-element analysis of patent infringement

310. *Alice* may alleviate concerns about non-practicing entities (often known as "patent trolls"), who, after buying software patents, often from failed startups, make demands for high licensing fees that some consider extortionate or bring lawsuits for patent infringement. *See, e.g*., James Bessen, *The Patent Troll Crisis Is Really a Software Patent Crisis*, WASH. POST (Sept. 3, 2013), https://www.washingtonpost.com/news/the-switch/wp/2013/09/03/the-patent-troll-crisis-is-really-a-software-patent-crisis/ [https://perma.cc/A9MA-92RA] (discussing rise in software patent litigation). *See generally* Colleen V. Chien, *Startups and Patent Troll*s, 17 STAN. TECH. L. REV. 461 (2014) (studying the distributional impacts of the demands of patent trolls).

311. *Alice*, 573 U.S. at 226–27 (2014) (discussing the Court's other recent patent subject matter cases, as well as *Benson* and *Flook*).

312. *See, e.g.*, Tran, *supra* note 138, at 533–34.

313. Samuelson, *supra* note 2, at 1497.

314. *Id.* at 1497–99.

315. *Id.* at 1497.

316. *Id.* at 1498.

claims.[317] Sixth, copyright remedies are more attractive for plaintiffs than patent remedies.[318] Disgorgement of a defendant's profits is recoverable in copyright, along with any lost profits (such as a license fee).[319] Patent law does not allow profit disgorgement at all, and because lost profits from infringement of a patent on a component of a larger manufacture may be difficult to prove, reasonable royalties are the most common patent infringement remedy.[320]

After *Alice*, many nonliteral elements of programs may be too abstract to qualify for patent protection. A copyright SSO claim may be plausible, though, as long as there is more than one way to structure those elements.[321] After *Alice*, issued patents on program SSO may be of questionable validity, so shifting one's claim to copyright as SSO would be a way to obtain exclusive rights. After all, every method or system that was patent subject matter under *State Street* would have some structure that could be renamed program SSO.[322] Software developers generally prefer to treat internal SSO as trade secrets instead of seeking patents.[323] But if competitors somehow manage to get access to that SSO, developers may well assert SSO copyright claims to stop its use in competing products. The temptation to assert copyright to fill in gaps in IP protections for software exists and is strong.[324]

Oracle provides strong incentives for software copyright plaintiffs to add a patent claim to their copyright complaints, for even if the patent claim is dismissed at an early stage of the case, any appeal of the copyright ruling will go to the CAFC, not to the regional circuits that would ordinarily hear appeals in copyright cases. Unless the Supreme Court decides to hear Google's latest appeal, the CAFC's *Oracle* decision may well enable aggressive software copyright plaintiffs to avoid application of the rigorous filtration tests previously adopted in *Altai* and

317. *Id.* at 1498–99.

318. *Id.* at 1499.

319. *Id.*; 17 U.S.C. § 504(a)–(b) (2012).

320. 35 U.S.C. § 284 (2012).

321. Proponents of broad software copyrights tend to ignore the role that patents play in the protection of software innovations. *See, e.g.*, Clapes et al., *supra* note 14, at 1558–60.

322. *See, e.g.*, Karjala, *Relative Roles*, *supra* note 17, at 66–68 (arguing that program SSO is patentable); Lemley, *supra* note 17 (asserting that the nonliteral elements of programs are patentable); John P. Sumner, *The Copyright/Patent Interface: Patent Protection for the Structure of Program Code*, 30 JURIMETRICS J. 107, 113–14 (1989) (stating that program SSO is patentable).

323. *See generally* David W. Carstens, *Legal Protection of Computer Software: Patents, Copyrights, and Trade Secrets*, 20 J. CONTEMP. L. 13 (1994) (discussing different forms of IP protection for software).

324. On December 14, 2016, at a Copyright Society of the U.S.A. debate in San Francisco on the *Oracle v. Google* case, Annette Hurst, who represented Oracle in that case, stated that IP lawyers are talking about the need for copyright to expand to make up for the lessened availability of patents. This is a bad idea. *See, e.g.*, Lemley, *supra* note 17, at 25–26 (arguing against expansive interpretations of copyright to fill gap if patents are unavailable).

its progeny, as well as to get exclusive rights in unpatentable methods and processes by claiming them as SSO.

But let us imagine what the future for software copyright law could look like if the Court does take Google's appeal and decides that *Altai* and its progeny were right: that copyright law should not be construed to confer patent-like protection to programs, that § 102(b)'s exclusions of methods and processes should be respected, and that it is for patent law to protect program functionality.

B. *How to Minimize the Potential Misuse of Copyright to Confer Patent-Like Protections for Nonliteral Elements of Software*

The first step toward minimizing the potential misuse of software copyright law is recognizing the risks to competition and innovation if the *Oracle* decision, in effect, resurrects *Whelan* and the patent-like protections it would enable simply by characterizing nonliteral elements as program SSO. The second step is for courts to do a better job of conceptualizing copyright protectable and unprotectable aspects of programs in keeping with the long-standing fundamental principle that copyright law should not provide patent-like protections to software or other functional innovations embodied in copyrighted works.

Over the past several decades, the only easily resolved software copyright cases have been those in which plaintiffs proved that defendants literally infringed source or object code or copied videogame audiovisuals or other expressive screen displays generated by programs.[325] Copyright provides very meaningful protection to these elements of programs. Thus, the resolution of these types of infringement claims has generated little or no controversy.

Nonliteral infringement claims have proven much more difficult for courts to resolve for several reasons. First, judges and juries typically lack the technical training necessary to comprehend programs' complex technological character.[326] Second, software is not a "literary work" in either the conventional sense of the term (for example, a novel) or the conventional copyright sense of the term (for example, a compilation of

325. *See e.g.*, Cadence Design Sys., Inc. v. Avant! Corp., 125 F.3d 824, 829 (9th Cir. 1997) (involving source code copying); Engenium Sols., Inc. v. Symphonic Techs., Inc., 924 F. Supp. 2d 757, 778–80 (S.D. Tex. 2013) (involving verbatim copying of code); Spry Fox LLC v. LOLApps Inc., No. 2:12-cv-00147-RAJ, 2012 WL 5290158, at *1, *8 (W.D. Wash. Sept. 18, 2012) (stating that there were enough similarities in videogame graphics to deny a defense motion to dismiss). Software copyrights might also be infringed by some nonliteral copying, as when a competitor recompiles another's code, directly translates the program from one programming language to another, or makes minor alterations (e.g., changing variable names) to disguise infringement.

326. *See, e.g.*, Nimmer et al., *supra* note 98, at 625–26.

data). It is a virtual machine that happens to have been constructed in text.[327] Copyright doctrines developed to assess infringement of novels and dramatic plays are ill-suited to informing judgments about software nonliteral infringements.[328] Third, while it is easy to say that copyright law protects only program expression and not program function, separating expression and function has, in practice, been maddeningly difficult.[329] Functional considerations pervade the design of virtually every aspect of software. Finally, courts have been surprisingly reluctant to construe and apply the § 102(b) method exclusions in software copyright cases or to recognize that the design of software largely lies in assembling functional compilations of applied knowhow.[330]

There are sound ways to address each of these problems. The Federal Judicial Center has organized training sessions to teach judges about software, other advanced technologies, and IP law.[331] Judges with experience in technology cases are now more frequently assigned to hear them.[332] Additionally, expert witnesses can provide important expertise to aid the trier of fact in complex technology cases, as the Second Circuit recognized in *Altai*.[333]

To overcome the misleading metaphor of software as a "literary work," courts should follow *Altai*'s lead in acknowledging that the essentially utilitarian nature of software means that programs have only

327. Samuelson et al., *supra* note 13.

328. *See, e.g.*, Weinreb, *supra* note 14, at 1151–53.

329. *See, e.g.*, Samuelson, *supra* note 8, at 1284–85.

330. *Altai*'s failure to direct filtering out processes and methods from the scope of copyright was likely due to the influence of the Nimmer treatise that, until recently, had an unduly narrow interpretation of § 102(b). *Id.* at 1235–37.

331. The Berkeley Center for Law & Technology, for instance, hosts an annual program for federal judges on intellectual property and technology law topics. *See Berkeley Center for Law & Technology*, RES. UC BERKELEY, http://vcresearch.berkeley.edu/research-unit/berkeley-center-law-and-technology [https://perma.cc/6SME-WLGV] (describing its annual judicial training sessions).

332. *See, e.g.*, MARGARET S. WILLIAMS ET AL., PATENT PILOT PORGRAM: FIVE-YEAR REPORT v (2016), https://www.fjc.gov/sites/default/files/2016/Patent%20Pilot%20Program%20Five-Year%20Report%20(2016).pdf [https://perma.cc/6897-U6E5].

333. Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 712–13 (2d Cir. 1992) (noting that "the highly complicated and technical subject matter at the heart of these claims" make them "somewhat impenetrable by lay observers" so expert testimony may aid resolution of the infringement issue). The Ninth Circuit has taken a more restrictive approach to expert testimony in software cases. *See* Antonick v. Elec. Arts, Inc., 841 F.3d 1062, 1067 (9th Cir. 2016). The Ninth Circuit was correct in ruling against Antonick because he claimed that Electronic Arts had copied his source code but produced no evidence of this. *See id.* at 1069. But the court's unwillingness to allow expert testimony in software cases except on the issue of probative similarities is questionable. *See, e.g.*, Brief of Amici Curiae Intellectual Prop. Law Professors in Support of Petitioner at 2, *Antonick*, 841 F.3d 1062 (No. 17-168).

a narrow scope of copyright protection.[334] Courts should also engage in rigorous filtering out of unprotectable elements before the comparison stage of infringement analysis, as *Altai* and its progeny direct.[335]

Altai offered important guidance about how to distinguish program expression and program function. By characterizing efficient designs as lying outside of copyright bounds under the merger doctrine and identifying several types of external factors that may constrain programmer design choices,[336] *Altai* offered useful tools with which to assess nonliteral infringement claims and filter out functional elements. This is similar to the process of claim construction that courts routinely employ in patent cases.[337] Just as those claiming trade dress protections for product configurations must prove their designs are nonfunctional,[338] plaintiffs asserting nonliteral infringement claims should have to prove that nonliteral elements of software alleged to infringe are neither efficient nor *scènes à faire*, nor otherwise constrained by external factors.[339] The existence of alternative ways to accomplish the function should not be dispositive of non-functionality.[340]

Unfortunately, *Altai* did not offer any guidance about how to filter out § 102(b) methods in software copyright cases. This is surprising given the explicit statement in the legislative history that § 102(b) method/process exclusions should ensure that software copyrights would not be construed too broadly.[341] Numerous cases have, however, adapted

334. *Altai*, 982 F.2d at 704.

335. *Id.* at 707.

336. *Id.* at 707–08.

337. *See, e.g.*, Mark A. Lemley & Mark P. McKenna, *Scope*, 57 WM. & MARY L. REV. 2197, 2202–03 (2016) (recommending that courts determine the scope of litigated IP rights in a manner akin to claim construction of patents).

338. *See* TrafFix Devices, Inc. v. Mktg. Displays, Inc., 532 U.S. 23, 29 (2001). It is currently unclear whether the plaintiff or defendant bears a burden of proof about non-functionality or whether particular elements of software are protectable expression or unprotectable methods. Some courts have put the burden of proof about unprotectables on defendants. *See, e.g.*, Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1541 (11th Cir. 1996). Especially when dealing with a highly functional work such as software, plaintiffs should have to prove what aspects of their programs are expressive enough to qualify for copyright protection. After all, plaintiffs bear the burden of proof that defendants improperly appropriated original expression from their works. *See, e.g.*, 3 GOLDSTEIN, *supra* note 276, § 9.3.2.2, at 9:46 ("The abstraction-filtration-comparison test places a special burden on the copyright owner to highlight the original and expressive elements that it claims are infringed.").

339. *See, e.g.*, Bikram's Yoga Coll. of Ind., L.P. v. Evolation Yoga, LLC, 803 F.3d 1032, 1040–41 (9th Cir. 2015); *Altai*, 982 F.2d at 709–10.

340. As Weinreb once observed, "if the rubric [of other possible design choices] is so used, copyright effectively absorbs the whole of patent." Weinreb, *supra* note 14, at 1170. The CAFC in *Oracle* concluded that the Java API was original expression because Sun/Oracle made choices among alternatives. Oracle Am., v. Google Inc., 750 F.3d 1339, 1356 (Fed. Cir. 2014).

341. *See supra* notes 52, 246 and accompanying text.

the AFC test to filter out § 102(b) methods in software copyright cases.[342] Commentators have also offered guidance about how to discern § 102(b) method exclusions by focusing on particular aspects of programs and using pertinent computer science terminology instead of vague terms such as "SSO."[343]

Among the aspects of programs that courts have excluded as unprotectable § 102(b) methods are the functional behavior of programs,[344] algorithms,[345] data structures,[346] command structures,[347] computer languages,[348] methods of calculation,[349] computational processes,[350] interfaces necessary to interoperability,[351] and formulae.[352] The Copyright Office *Compendium* has identified numerous aspects of programs that it regards as unprotectable by copyright, including algorithms, languages, interfaces, formulae, logic, system design, and formats.[353]

Since *Altai*, courts have generally required software plaintiffs to particularize the elements of computer programs on which they base their nonliteral infringement claims.[354] Courts also typically engage in

342. *See, e.g.*, *Bateman*, 79 F.3d at 1541 n.21; Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 842–43 (10th Cir. 1993) (excluding algorithms from software copyright scope). Patented elements should also be filtered out. *See, e.g.*, Lemley, *supra* note 17, at 32.

343. *See, e.g.*, Marci A. Hamilton & Ted Sabety, *Computer Science Concepts in Copyright Cases: The Path to a Coherent Law*, 10 HARV. J.L. & TECH. 239, 240 (1997); *see also* Randall Davis, *The Nature of Software and Its Consequences for Establishing and Evaluating Similarity*, 5 SOFTWARE L.J. 299, 301–09 (1992) (computer scientist's explanation of program components).

344. *See, e.g.*, O.P. Sols., Inc. v. Intellectual Prop. Network Ltd., No. 96 Civ. 7952 (LAP), 1999 WL 47191, at *16–19 (S.D.N.Y. Feb. 2, 1999).

345. *See, e.g.*, Torah Soft Ltd. v. Drosnin, 136 F. Supp. 2d 276, 291 (S.D.N.Y. 2001).

346. *See, e.g.*, Baystate Techs., Inc. v. Bentley Sys., Inc., 946 F. Supp. 1079, 1088–89 (D. Mass. 1996); *see also* Hamilton & Sabety, *supra* note 343, at 259–64 (observing that a programmer's choice of algorithms may constrain data structure choices).

347. *See, e.g.*, MiTek Holdings, Inc. v. Arce Eng'g. Co., 89 F.3d 1548, 1556–57 (11th Cir. 1996); Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996). *But see* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1356–59 (Fed. Cir. 2014). Some cases deny similar copyright claims on other grounds. *See, e.g.*, Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366, 1373–76 (10th Cir. 1997) (concluding Mitel's command codes were unprotectable on *scènes à faire* and unoriginality grounds).

348. *See, e.g.*, SAS Inst. Inc., v. World Programming Ltd., 64 F. Supp. 3d 755, 776 (E.D.N.C. 2014), *aff'd in part, vacated in part*, 874 F.3d 370 (4th Cir. 2017).

349. *See, e.g.*, Harbor Software, Inc. v. Applied Sys., Inc., 925 F. Supp. 1042, 1052 (S.D.N.Y. 1996).

350. *See, e.g.*, Wyatt Tech. Corp. v. Malvern Instruments, Inc., No. CV 07-08298 DDP (manX), 2009 WL 2365647, at *6 (C.D. Cal. July 29, 2009).

351. *See, e.g.*, Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1522–23 (9th Cir. 1992).

352. *See, e.g.*, Woods v. Resnick, 725 F. Supp. 2d 809, 820 (W.D. Wisc. 2010).

353. U.S. COPYRIGHT OFFICE, *supra* note 44, § 721.7.

354. *See* MiTek Holdings, Inc. v. Arce Eng'g. Co., 89 F.3d 1548, 1555 (11th Cir. 1996) (limiting the infringement analysis to eighteen specific similarities).

element-by-element analysis to determine whether the nonliteral elements allegedly infringed are copyright-protectable or unprotectable.[355] Plaintiffs' failure to particularize elements alleged to be infringed and trial courts' failure to filter out unprotectables have resulted in some summary judgments for defendants and some reversals of plaintiff victories.[356]

Plaintiffs in software copyright cases may prefer to employ vague terms such as SSO to describe the nonliteral elements alleged to infringe because such terms obscure, rather than illuminate, which program structures are expressive enough to be copyright-protectable and which are not.[357] Courts should consequently be much more skeptical about SSO claims than they have been to date. All procedures, processes, systems and methods of operation are, by their nature, parts of program SSO. Section 102(b) directs that these elements be excluded from the scope of copyright.[358] Moreover, as *Altai* pointed out, the term SSO "demonstrate[s] a flawed understanding of a computer program's method of operation" and rests on a "somewhat outdated appreciation of computer science."[359] Plaintiffs should be required to prove the expressiveness of structural elements they allege as the basis of nonliteral infringement claims. Courts should also recognize the benefits of standardization in software copyright nonliteral infringement cases.[360]

Defenses that rely on issued patents on program methods/systems should be treated as evidence that these elements are uncopyrightable under § 102(b).[361] Although fewer software-related patents may have been issued since *Alice*, hundreds of thousands of software patents still

355. *See, e.g.*, *id.*

356. *See* Automated Sols. Corp. v. Paragon Data Sys., 756 F.3d 504, 520–21 (6th Cir. 2014) (affirming summary judgment because plaintiff failed to specify elements alleged to infringe); *see also* Paycom Payroll, LLC v. Richison, 758 F.3d 1198, 1207–08 (10th Cir. 2014) (vacating a ruling for plaintiff due to failure to filter out unprotectable elements).

357. *See, e.g.*, Chisum et al., *supra* note 18, at 20–21.

358. 17 U.S.C. § 102(b) (2012).

359. Computer Assocs. Int'l v. Altai, Inc., 982 F.2d 693, 706 (2d Cir. 1992); *see also* Hamilton & Sabety, *supra* note 343 (stating that SSO does not "accurately reflect[] computer science reality").

360. *See, e.g.*, Joseph Farrell, *Standardization and Intellectual Property*, 30 JURIMETRICS J. 35, 42–43, 47, 49 (1989); Weinreb, *supra* note 14, at 1204 (discussing standardization in relation to *Borland*).

361. *See supra* Section III.C; *see also* NIMMER & NIMMER 2018, *supra* note 17, § 2A.07[D][4][c][iii] ("[A] court adjudicating a case presenting the utility patent/copyright overlap must calibrate copyright protection with reference to Section 102(b). . . . [I]f the court concludes that plaintiff can vault the Section 102(b) hurdle, then the existence of patent protection should cause the court to re-evaluate its conclusion about Section 102(b).").

exist.[362] Because *Alice* may undermine the enforceability of such patents, developers may claim them as program SSO instead. Copyright law should not be used as a substitute or gap-filler for programmers' failure to get or to seek a patent for functional design elements of programs or for patents of dubious validity.[363]

Courts should further be skeptical about compilation claims in software cases.[364] The "industrial compilation[s] of applied know-how" in programs should generally be ineligible for copyright protection.[365] *Microsoft* rightly held that the selection and arrangement of unprotectable elements of the Apple GUI were too functional to be copyright-protectable.[366] Only when software elements have been selected and arranged in an expressive way should copyright protection be available to a compilation of them.[367] The existence of alternative ways to organize the elements may be a factor in assessing whether the creativity required

362. *See* Raymond Millien, Alice *Who? Over Half the U.S. Utility Patents Issued Annually are Software Related!*, IPWATCHDOG (May 21, 2017), https://www.ipwatchdog.com/2017/05/21/alice-over-half-u-s-utility-patents-issued-annually-software/id=83367/ [https://perma.cc/A743-QFBR].

363. If a plaintiff received a patent on a software design (e.g., a data structure) and then claimed copyright in it, relying on the overlap dicta in *Oracle*, a court should decide that the plaintiff's election of patent protection precludes copyright in it. *See, e.g.*, Korzybski v. Underwood & Underwood, Inc., 36 F.2d 727, 729 (2d Cir. 1929) (finding that a patent on an anthropometer design precluded a claim of copyright infringement); *see also* Michael J. Kline, *Requiring an Election of Protection for Patentable/Copyrightable Computer Programs*, 6 COMPUTER L.J. 607, 609 (1986) (suggesting that a programmer should be required to choose either patent protection or copyright protection when a program qualifies for both).

364. *Cisco v. Arista* illustrates the misuse of compilation theory. Cisco claimed that 508 command-line interface terms Arista used from as many as twenty-six Cisco programs was a compilation whose copyright Arista infringed. Cisco Sys., Inc. v. Arista Networks, Inc., No. 14-cv-05344-BLF, 2016 WL 4440239, at *4 (N.D. Cal. Aug. 23, 2016). This so-called compilation apparently didn't exist before Cisco filed suit. *Id.* at *3–4. At trial, Arista's *scènes à faire* defense succeeded. Order Denying Motions for Judgment as a Matter of Law & Motion for a New Trial, Cisco Sys., Inc. v. Arista Networks, Inc., Case No. 14-cv-05344-BLF, 2016 WL 4440239 (N.D. Cal. May 10, 2017); *see also* eScholar LLC v. Otis Educ. Sys. Inc., 76 U.S.P.Q.2d (BNA) 1880, 1898 (S.D.N.Y. 2005) (relying on compilation theory to extend protection to program methods).

365. Samuelson et al., *supra* note 13, at 2355.

366. Apple Comput., Inc. v. Microsoft Corp., 799 F. Supp. 1006, 1023 (N.D. Cal. 1992); *see supra* note 183 and accompanying text; *see also* MiTek Holdings, Inc. v. Arce Eng'g. Co., 89 F.3d 1548, 1558–59 (11th Cir. 1996) (adopting the "bodily appropriation of expression" or "virtual identicality" standard used in *Microsoft*). Many non-software compilations have been denied copyright protection because they were too functional to be copyrightable. *See* Pamela Samuelson, *Functional Compilations*, 54 HOUS. L. REV. 321, 323–54 (2016) (identifying several types of functional compilations held unprotectable by copyright law).

367. 17 U.S.C. § 101 (2012) (definition of compilation); *see also* Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 362 (1991) (discussing the quantum of originality required for compilations to qualify for copyright).

to select and arrange the elements is of the sort that copyright law was intended to protect, but it should not be dispositive.

Nonliteral infringement software copyright cases could be much easier if courts decided that copyright protection extends only to program code and expressive screen displays, as Rand Jaslow argued many years ago in *Whelan*.[368] However, the proposition that software copyrights can be nonliterally infringed is now so well established that it seems unlikely courts will shift in that direction, however persuasive the arguments supporting that approach may be.[369]

Courts can and should be very rigorous in their infringement analyses in software copyright cases so that plaintiffs cannot get patent-like protection for functional structures of programs or protection for other unprotectable elements. Competition and ongoing innovation policy, as well as the progress of knowledge, will be fostered if programmers can draw upon uncopyrightable components of programs in constructing new software.[370]

CONCLUSION

Congress was warned as early as 1967 that copyright protection for computer programs might result in conferring patent-like protection to functional aspects of programs, such as the methods and processes they embody. Congress did the right thing in responding to this concern by adding § 102(b) to the 1976 Act hoping to ensure that "in no case" would copyright protection extend to any "procedure, process, system, [or] method of operation" of programs.[371] The House and Senate reports plainly reflect the intention that these exclusions would stop program copyrights from being construed too broadly.

The § 102(b) functionality exclusions were ignored in *Whelan* and its progeny in the mid-1980s to early 1990s, which led to a storm of criticism from IP lawyers and academics, as well as to some alarm by computing professionals. Responding to these concerns, Congress once again did the right thing in the late 1980s and early 1990s by holding hearings and

368. *See* Whelan Assocs., Inc. v. Jaslow Dental Labs., Inc., 797 F.2d 1222, 1235 (3d Cir. 1986).

369. *See* Karjala, *Relative Roles*, *supra* note 17, at 52–53; *see also* Weinreb, *supra* note 14, at 1249–50 (arguing against nonliteral infringement); Goldstein, *supra* note 17, at 1125 (stating copyright provides "very thin" protection to programs).

370. *See, e.g.*, Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141, 156–57 (1989) ("Both the novelty and the nonobviousness requirements . . . provide the baseline of free competition upon which the patent system's incentive to creative effort depends. A . . . law that substantially interferes with the enjoyment of an unpatented utilitarian or design conception which has been freely disclosed by its author to the public at large impermissibly contravenes the ultimate goal of public disclosure and use which is the centerpiece of federal patent policy.").

371. 17 U.S.C. § 102(b).

commissioning reports to investigate the state of software copyright and patent protection for programs and what Congress might do about it.

The turning point away from *Whelan*-like overprotection was the Second Circuit's *Altai* decision, which recognized the highly utilitarian nature of programs and directed courts to engage in rigorous filtration to prevent copyright protection from extending to program functionality. As a result of this decision and others that followed its lead, the risks of harm to competition and innovation in the software industry from overbroad copyright protections subsided.

For more than twenty years, courts following *Altai*'s lead were mindful that copyright and patent have different roles to play in protecting software, and the § 102(b) exclusions of methods and processes, the merger doctrine, and the explanation/use distinction have served as useful tools with which to maintain proper boundaries between these two forms of exclusive rights. As long as courts have kept in mind the highly utilitarian nature of software and recognized that protecting functionality is the appropriate role of patents, they have generally done a good job steering a course through the Scylla of underprotection and the Charybdis of overprotection. When courts interpret software copyright in the shadow of patents, they are less likely to exceed the boundaries that Congress intended to establish by enacting § 102(b).

By misconstruing *Mazer, Altai*, and § 102(b), the *Oracle* decision has reignited concerns about copyright law being construed to provide patent-like protections to programs. The decision has already precipitated a new round of software copyright cases seeking *Whelan*-like protection of nonliteral elements as SSO. The CAFC's endorsement of copyright and patent overlaps is inconsistent with its own and many other precedents, as well as with longstanding limiting principles of copyright law. How tempting it is now for software developers to claim SSO copyright protection for unpatented functionality. And how tempting to add a patent claim to a software copyright complaint to ensure that any appeal will go to the CAFC where they can rely on its *Oracle* precedent, thereby avoiding the regional circuits that would have followed the *Altai*-plus rigorous filtration precedents. Congress is too busy with other matters to save the day, and not even the Second or Ninth Circuits can fix the problem that the *Oracle* decision has created. If the Supreme Court decides to hear Google's appeal of the CAFC's *Oracle* decision, courts may get the guidance they need to construe the scope of copyright protection for software in the manner Congress intended back in the 1980s.